Parallelizing *K-means* with Hadoop/Mahout for Big Data Analytics

A Thesis Submitted for the Degree of

Master of Philosophy

By

Jianbin Cui

Department of Electronic and Computer Engineering

College of Engineering, Design and Physical Sciences

Brunel University, UK

June 2015

Abstract

The rapid development of Internet and cloud computing technologies has led to explosive generation and processing of huge amounts of data. The ever increasing data volumes bring great values to societies, but in the meantime bring forward a number of challenges. Data mining techniques have been widely used in decision analysis in financial, medical, management, business and many other fields. However, how to analyse and mine valuable information from the massive data has become a crucial problem as the traditional methods are hardly to achieve high scalability in data processing.

Recently, MapReduce has emerged into a major programming model in dealing with big data analytics. Apache Hadoop, which is an open-source implementation of MapReduce, has been widely taken up by the community. Hadoop facilitates the utilization of a large number of inexpensive commodity computers. In addition, Hadoop provides support in dealing with faults which is especially useful for long running jobs. Mahout is a new open-source project of Apache, providing a number of machine learning and data mining algorithms based on the Hadoop platform.

As a machine learning technique, K-means has been widely used in data analytics through clustering. However, K-means experiences high overhead in computation when the size of data to be analysed is large. This thesis parallelizes K-means using the MapReduce model and implements a parallel K-means with Mahout on the Hadoop platform. The parallel K-means reduces the computation time significantly in comparison with the standard K-means in dealing with a large data set. In addition, this thesis further evaluates the impact of Hadoop parameters on the performance of the Hadoop framework.

Keywords: *cloud computing, big data, Hadoop, Data mining, K-means*

DECLARATION

The work in this thesis is based upon the research carried out at the Department of Electronic and Computer Engineering, Brunel University. Except where specific reference has been made to the work of others, this thesis is the result of my own work. No part of this thesis has been submitted elsewhere for any other degree or qualification.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my supervisors Professor Maozhen.Li and Dr. Hongying Meng for their support, encouragement, guidance and patience during the research period. I am deeply grateful of their help in the completion of this thesis. Their valuable guidance and suggestion help me through the hardest times.

I would like to thank my family members and my friends. Without their help and trust, I will not finish the work successfully. I am also deeply indebted to all the colleagues in university for their direct or indirect help to me.

Table of Content

DECLARATION i

Chapter 1 Introduction 1

1.1 Background 1

1.2 Cloud Computing and Data Mining 4

1.3 Major Contributions 8

1.4 Thesis Structure 9

Chapter 2 A Review on Hadoop/MapReduce 10

2.1 Hadoop Overview 10

2.1.1 Hadoop History 10

2.1.2 Hadoop Advantages 10

2.1.3 Hadoop Subprojects 11

2.2 MapReduce Programming Model 12

2.2.1 MapReduce Logical Model 12

2.2.2 MapReduce Execution Flow 13

2.2.3 MapReduce Fault-tolerant Mechanism 15

2.3 HDFS Mechanism 16

2.3.1 HDFS Features and Limitations 16

2.3.2 HDFS Related Concepts 17

2.3.3 HDFS Framework 18

2.4 Mahout Algorithm Library 21

2.4.1 Introduction 21

2.4.2 Taste 22

2.4.3 Mahout Algorithm 23

2.5 Summary 24

Chapter 3 The Parallel Strategy of K-means Algorithm 25

3.1 Clustering Analysis 25

3.2 K-means Algorithm Descriptions 26

3.3 K-means Algorithm Parallel Strategy 30

3.3.1 Mahout Data Model 30

3.3.2 K-means Parallel Design Based on MapReduce 41

3.4 Summary 45
Chapter 4 Experimentation and Evaluation 46
4.1 Building experimental platform 46
4.1.1 Software Environment 46
4.1.2 Set up Hadoop Platform 47
4.2 Experimental Studies 48
4.2.1 Efficiency of K-means Algorithm in Serial and Parallel Environment 48
4.2.2 Optimization of Hadoop Parameter 56
4.3 Summary 61
Chapter 5 Conclusions and Future Work 62
5.1 Conclusions 62
5.2 Future Work 62
References 64

List of Figures

Figure 2.1: Data flow diagram of WordCount 13
Figure 2.2: MapReduce program execution flow 14
Figure 2.3: HDFS internal framework 19
Figure 2.4: Core components of Taste 22
Figure 3.1: Result of first clustering 28
Figure 3.2: Result of second clustering 29
Figure 3.3: Comparison between Euclidean Distance and Manhattan Distance 33
Figure 3.4: Flow chart of once iteration 41
Figure 4.1: Hadoop platform working status 47
Figure 4.2: Nodes working status 48
Figure 4.3: Output folder 50
Figure 4.4: Result demonstration 51
Figure 4.5: Result folder 52
Figure 4.6: Weka working status 53
Figure 4.7: Result of parallel environment 54
Figure 4.8: Efficiency between single node and serial environment 55
Figure 4.9: The impacts of io.sort.factor 59
Figure 4.10: The impacts of number of reduce tasks 60
Figure 4.11: The impacts of Java_opts value 60

List of Tables

Table 2.1: Mahout algorithm library..... 23
Table 3.1: Orange Vectors 35
Table 3.2: The number of words..... 37
Table 4.1: Environmental configuration of experiment..... 46
Table 4.2: Data information..... 49
Table 4.3: Experimental results 54
Table 4.4: Parameters of Hadoop 56

Chapter 1

Introduction

1.1 Background

With the rapid development of mobile Internet and e-commerce, the Internet has entered a new era. The amount of data generated has expanded sharply. For example, New York Stock Exchange generates 1TB of transactions daily, Facebook stores around 10 billion photos and the storage capacity is about 1PB, the Large Hadron Collider near Geneva, Switzerland produces about 15 PB of data annually [1]. Some Internet service providers face enormous challenges. The traditional data processing and the analysis framework are incapable of storing the large-scale data efficiently and supporting the high throughput literacy tasks. According to an investigation from IDC, the global amount of digital data doubles every two years, the amount of data had reached 1.8 ZB at the end of 2011 [2]. The transformation of information usage patterns for business and personal user has far exceeded the limits of the existing system platforms. The amount of data processed is overwhelming for enterprise and individual users.

Large amounts of data could be produced due to the application of measurement techniques and equipment in the science and engineering exploration, which would lead to huge challenges for processing large-scale massive data. For scientists and engineers, it is necessary to analyse the massive data in order to discover some useful information, but it is very difficult [3]. Due to the rapid development of genetics, people already have the conditions for conducting the exploration of hidden inheritance. However, it is a challenge to sort the genetic data, match and further mine the useful information. In the field of medical research, scientists have collected a plenty of diagnostic records of patients, physiological information and medical test

data in order to extract the useful information. In the field of meteorological research, the researchers have collected the meteorological environmental information of recent hundreds years. However, due to the lack of massive data processing technology, it is difficult to analyse the massive meteorological data. In the era of artificial intelligence, it is required that the machine has the ability to store the large-scale experience and knowledge, then extract the knowledge for learning and logical reasoning from the data and provide the expected feedback. The same problem exists in the field of humanities, such as huge amount of data is produced in the social network involving interpersonal and social relations.

With the increasing popularity of the sensor, the way to collect the information has become more diversified. The variety of sensors used by Real-time monitoring and network traffic monitoring system will generate different types of data [4]. Besides, the streaming data now also exists in different fields, such as the communication data in communication fields, the transactions in retail fields, the stock prices and transactions in financial sector. The streaming data is orderly and changing rapidly, which result in higher demand for response speed when processing and analyzing of streaming data.

The reasonable usage of massive data has become a popular and practical research. The data that we studied is called 'Big Data' and has three features: volume, velocity and variety. The information of the plentiful, complex structure data has many differences with the data in common databases. This series of development increase the difficulty of using data mining technology. The large-scale computing ability becomes an important external demand for the data mining technology to effectively complete the job. Traditionally, enterprises will attempt to use the high-performance computer or large-scale computing equipment to calculate the data, but it will greatly increase the cost of implementation. In order to adapt the current trends, cloud computing is presented to the stage of computer science and application, it not only leads a big change of information usage patterns, but also proposes a new way for data mining. Cloud computing is a new platform which focuses on big data and

distributed parallel processing, developed rapidly in recent years and achieved initial success in business. In the era of cloud computing, the traditional clustering algorithms can be redesign and implement using cloud computing platform in order to reduce the time and space complexity and efficiently solve the bottleneck problems in the storage and computing of big data [5].

Although the big data has features of being giant and chaotic, after professional process, it is able to get in-depth information with more decision-making power. Clustering is dividing a data set into a number of clusters or classes, and make the data objects in the same cluster have high similarity. The data objects in the different clusters are not similar. Because of the huge amount of data, especially in the Internet field, using traditional clustering algorithms to analysis big data will cost numerous time and memory space while unable to meet the request of real-time response. Hadoop is an open-source cloud computing platform with excellent data storage and computing performance to overcome the traditional bottleneck when processing big data. Currently, Hadoop is the preferred way to analyse big data. Meanwhile, Mahout is one of Hadoop subprojects, which is designed and implemented based on Hadoop platform. It focuses on the algorithms of data mining and machine learning, which is a perfect choice to be studied as a clustering algorithm in the cloud platform.

In recent years, data mining has aroused great concern of the information industry, which is mainly due to large amounts of data generated every day and converted into useful information and knowledge pressingly. The acquired information and knowledge can be used for market analysis, fraud checking, customer retention, product control and many other aspects of scientific inquiry [6].

Data mining is the basis of artificial intelligence, machine learning, pattern recognition, statistical, database, visualization technology. It can highly automatically analyse the data of enterprise, make inductive reasoning, and then mine potential models to help decision-makers to adjust marketing strategy in order to reduce the risk and make the correct decisions. Data mining tasks including cluster analysis, classification analysis, correlation analysis and anomaly analysis.

The data mining development has been through the following stages: The first stage is e-mail. This stage can be considered as from the beginning of the 1970s, the average annual growth of communication traffic increased several times. The second stage is information release. Since 1995, the information release system with web technology as the representative explosively grew up and became the major Internet applications. The third stage is e-commerce. E-commerce is regarded as a sign of The Times, which is because the main commercial application of Internet is e-commerce. In the era of e-commerce, more transactions were completed via the Internet in order to make the transactions easier, the information also showed explosive growth. The fourth stage is full e-commerce. With the occurrence of SaaS (Software as a Service) pattern, software has access to the Internet, extend the e-commerce chain, and then form a new pattern concept called 'full e-commerce'. Nowadays, increasing enterprises are using data mining to preferably grasp the market and user behaviour so that their sales performance can be improved. The typical examples such as Amazon and other e-commerce websites increase sale performance through personalized recommendations. Bank conducts customer management and risk assessment by establishing data analysis models. With the advent of era of big data, data mining will certainly show an unprecedented vitality [7].

1.2 Cloud Computing and Data Mining

Cloud computing is the development of parallel computing, distributed computing and grid computing. It is a result of mixing evolution and sublimation some concepts which included virtualization, utility computing, IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service). In technical aspect it has a higher level of maturity, and is actively promoted by some large companies. Therefore, cloud computing developed extremely fast once proposed. Google, Amazon, IBM, Microsoft, Yahoo and some other companies are the cloud computing technology leaders and guiders.

Google is the largest user of cloud computing technology. Google search engine is

distributed in a large cluster including more than 200 sites and 1 million servers as well as the number of its server infrastructures are increasing promptly. Some of the successful applications are using cloud computing technology, such as Google earth, map, Gmail, Docs. Their data will be stored in a location on the Internet, and the user can use any device to connect to the Internet, which is a very convenient way to access and share the data. Currently, Google has opened its cloud computing facility to third-parties. Users can run large parallel application by using Google App Engine [8]. In 2004, it had published three papers to demonstrate the results of its cloud computing: GFS [9], MapReduce [10], Bigtable [11], and then opened programming courses in some universities in United States and China to lecture how to develop cloud computing. Thereafter, there are numerous followers. The most popular open-source project is Hadoop. Apache Hadoop is an open-source project on cloud computing, which allows distributed processing big data by using MapReduce computing model on a large number of cheap clusters. Transformation has been made from relying on expensive hardware into using distributed storage and parallel computing on cheap nodes to obtain high availability. Furthermore, Hadoop can detect and solve the node failures problem in order to provide high availability services in the case of individual nodes failed in the cluster. MapReduce model depends on Hadoop Distributed File System (HDFS) which support the local storage and computing of cluster nodes. Apache Mahout is an open-source algorithm library in data mining and machine learning fields, these algorithms are based on MapReduce programming model and HDFS.

Amazon developed Elastic Computing Cloud (EC2) and Simple Storage Service (S3) to provide computing and storage services to enterprises. It provides storage space, bandwidth, CPU resources and other rechargeable projects. Storage space and bandwidth charges by volume, users can store any type of file temporarily or permanently on S3 servers through the service interfaces which are provided by Amazon.

IBM introduced the 'Blue Cloud' computing platform in November 2007, aims to

provide the users with a 'Ready to use' cloud computing platform. It is composed of a series of self-management and self-healing virtualization cloud computing software components, users in the different places in the world are able to easily use the distributed large-scale server computing nodes. Afterwards, IBM is in cooperation with 17 European organizations had presented a RESERVOIR cloud computing project to achieve the 'Resources and Services Virtualization without barriers', the EU provided 170 million Euro to sponsor the project. In August 2008, IBM announced that it would invest about $ 400 million to transform its cloud computing data centre which are located in North Carolina and Tokyo Japan, and invest $ 300 million to build 13 cloud computing centres in 10 countries in 2009.

In the cloud computing era, Microsoft also issued a cloud computing system named 'Windows Azure' in October 2008, which focuses on developing terminal products. Azure is based on Internet architecture to create new cloud computing platform, which is another important strategic product after the Microsoft Windows replaced DOS. The underlying of Windows Azure cloud computing platform is Microsoft Global Foundation Services System, which is composed of the fourth generation of data centres distributed around the world. Windows Azure cloud computing platform provides infrastructure that can be accessed via the Internet, including processors, storage devices and services, users and companies also can run their applications and store data on Microsoft cloud computing platform. Currently, Microsoft has configured 220 containerized data centres with 440,000 servers.

Hadoop is an open-source project from Apache Foundation which achieves Google's GFS and MapReduce. In 2004, Doug Cutting and Mike Cafarella realized Hadoop Distributed File System and MapReduce distributed computing framework as well as published the original version. In December 2005, Hadoop can stably run on a cluster of 20 nodes. In January 2006, Doug Cutting joined Yahoo, the same year in February, Apache Hadoop project officially supported HDFS and MapReduce independent development. Meanwhile, the new company named 'Cloudera' provides commercial support for Hadoop to help enterprises realizing the standardize installation and

volunteer to contribute the community. Hadoop provides a distributed system infrastructure for developers, users can develop distributed applications to leverage the function of clusters and achieve high-speed computing and storage even if they do not understand the underlying details of distributed system. Due to the prominent advantages of Hadoop, the Hadoop-based applications have been very abundant, especially in the Internet field.

In 1967, Q.J.Mac proposed K-means clustering algorithm which has become one of the most popular clustering algorithms currently [12]. After that, many researchers made plenty of improved algorithms to overcome the disadvantages of K-means algorithm. Aristidis Likas with his group proposed a global K-means algorithm, which firstly initiates all data as a cluster centre, then tried each data in the dataset as the cluster centre and calculate the error, and then choose the data with the smallest error as the new cluster centre [13]. The process is repeated until the K centres are found. This algorithm can overcome the uncertainty which is brought from the random selection k centres, but the algorithm process requires a large amount of computation and more time. It is therefore not suitable for large amounts of data clustering analysis. Steinbach and his group proposed bisecting K-means algorithm, which firstly randomly selects two points, execute basic K-means algorithm, and then calculates the squared error, find the cluster with larger squared error. Two initial centres are selected randomly. This procedure is performed until the K cluster centres are found. Dan Pelleg and his group proposed x-means algorithm [14]. Firstly, the user specified one range for the K value, the algorithm initialized K to the minimum value in the range, executing the basis K-means algorithm. Afterwards, each cluster is divided into two clusters. Bayesian Information Criterion (BIS) is used to compare the result after divided with the original result. If the value of BIS becomes bigger, then accept the new division. It will find the most suitable K value. The stand-alone version of the x-means algorithm had been implemented in open source data mining algorithm tools named 'Weka'.

Nowadays, the attention of Hadoop-based algorithm increases sharply. There are

already a number of studies regarding to clustering algorithms based on MapReduce framework. Meanwhile, there are studies of K-means algorithm based on large-scale TCPdump dataset [15], studies of Jarvis-Patrick algorithm parallel research based on clustering [16], studies of improved algorithm Cop-Kmeans base on MapReduce platform [17], studies of ISODATA algorithm base on remote sensing image dataset implemented on the MapReduce [18]. Besides, there are few studies of research and test clustering algorithm based on the Mahout algorithm [19]. These papers tested scalability of K-means clustering algorithm in dataset and the size of clusters based on the Mahout platform.

The conducted research indicated that clustering algorithm based on Hadoop platform has lower time and space complexity and good scalability. A number of research works are conducted on Hadoop-based clustering algorithm and testing, however, there are few parallel research and test based on the Mahout platform. It is necessary to use a series of outstanding Mahout-based data model to do further research and testing.

1.3 Major Contributions

This thesis aims to research data mining algorithms in Mahout subprojects based on Hadoop, including the following aspects:

1) Firstly, analysis is conducted on the MapReduce programming model and the mechanism of Hadoop distributed file system based on Hadoop distributed computing platform.

2) Secondly, discussion is synthetically carried out on clustering algorithms in data mining algorithms and analyses the K-means algorithm process.

3) Finally, the Mahout framework is studied including the data representation in Mahout. K-means algorithm is used as an example to understand how to implement data mining algorithms in parallel based on Hadoop platform. Experimental study has been carried out to investigate the impact of different factors on the parallel

computing. The working efficiency of K-means algorithm is compared between serial environment and parallel environment.

1.4 Thesis Structure

This thesis is divided into the following five parts:

The first chapter is the introduction which aims to introduce the background and significance of this topic. Meanwhile, it introduces the current development situation of cloud computing and data mining. In the end, it puts forward the main contents of research.

The second chapter aims to research and analyse Hadoop technology framework including the most significant two sub-projects, which are MapReduce programming framework and Hadoop distributed file system. The Mahout algorithm library is introduced, which realizes machine learning and data mining based on Hadoop platform.

The third chapter focuses on introducing the clustering analysis of data mining and describing the characteristics of K-means algorithm. Further research is conducted about the data representation and parallel implementation of K-means clustering algorithm in the Mahout framework.

In the fourth chapter, the Hadoop platform and experimental environment are built, using the Reuters news collection and some other dataset to do K-means algorithm experiments. Different amounts of data are adopted to run the K-means algorithm separately in serial and parallel models. The efficiency between the two results is compared.

In the fifth chapter, the main research achievements of the thesis are summarized. The insufficient during the research work is analysed. The prospect of further work in the future is proposed.

Chapter 2

A Review on Hadoop/MapReduce

2.1 Hadoop Overview

Hadoop is an open-source Cloud Computing platform in Apache Foundation, which is based on MapReduce computing model and HDFS distributed system [1]. The advantages of HDFS such as high fault tolerance, high scalability, et al., enable it to deploy Hadoop distributed system on a large number of cheap clusters. Meanwhile, MapReduce supports the users to develop parallel, distributed application, and they do not need to understand the underlying details. Then, users can deploy Hadoop framework to utilize the resource advantages of large-scale clusters, in order to solve the big data analysis problem which is difficult for the traditional high-performance single machine [20].

2.1.1 Hadoop History

Hadoop comes from the Apache Nutch, which is a project that began in 2002. It is one of the Apache Lucene subproject. In 2004, Google published a thesis named ‘MapReduce: Simplified Data Processing on Large Cluster’ in ‘Operation System Design and Implementation’ conference. After that, Doug Cutting, who began to try to program MapReduce framework and combine it with NDFS, to support the main algorithms of Nutch. Due to the advantages of NDFS and MapReduce, they were separated out in February 2006 and had become a complete and independent project called Hadoop [21].

2.1.2 Hadoop Advantages

Hadoop is a distributed computing platform that users can easily build and use. There

are some main advantages including [22]:

- *High reliability*: Hadoop has high bitwise storage and processing data ability.
- *High scalability*: Hadoop distributes the data to the computers in the cluster, and completes the computing tasks. It is easily extended to thousands of nodes.
- *High efficiency*: Hadoop could dynamically move data among nodes, and ensure each node's dynamic equilibrium. So its processing speed is extremely fast.
- *High fault tolerance*: Hadoop can automatically store multiple same copies of the data as well as automatically reassign the failed task.

2.1.3 Hadoop Subprojects

Hadoop has become a set including multiple subprojects. Although its core is HDFS and MapReduce, the Hive, Common, Avro, Chukwa et al. subprojects are also indispensable. They provide complementary services or provide a higher level service in the core layer.

Core/Common: From Hadoop 0.20 version, Hadoop Core changed the name to Common. Common is a tool to support Hadoop subprojects, including File System, RPC and Serialization library.

Avro: Avro is an efficient data serialization system that provides data persistence storage and remotes procedure call RPC.

Chukwa: Chukwa is an open-source data collection system that is used for monitoring and analysing large-scale distributed system data.

Hive: Hive is a data warehouse to provide data summary and query function which was first designed by Facebook.

HBase: HBase is an open-source, distributed, column-oriented store modelled after Big Table.

Pig: Pig is a kind of advanced streaming language based on MapReduce.

Mahout: Mahout is based on Hadoop platform and realizes a variety of machine learning and data mining algorithms library.

Then, my work focuses on the main components of Hadoop: MapReduce and HDFS.

2.2 MapReduce Programming Model

MapReduce is an easily programming model used for data processing with fast realization and strong scalability. It not only uses Ruby, Python, PHP, C++ and some other non-java language to write in Map or Reduce program, but also runs the same program in any Hadoop cluster synchronously. Therefore, the key advantage of MapReduce is processing the big data set.

2.2.1 MapReduce Logical Model

There are two stages (map and reduce) during the MapReduce working process. Map is separating a task into multiple tasks and Reduce is for collecting and analysing results [23]. Map stage has function: map () and Reduce stage has function: reduce (). Each function regards a key-value pair as input and output, in which ‘key’ means the byte offset in each data record in a data slice, and ‘value’ is the content of each row [24].

There is a basic requirement suitable for MapReduce processing tasks: Data sets can be processed into numerous small data sets, and every small data set can be processed completely in parallel. There is a ‘WordCount’ example to analyse the whole idea of the programming model.

Setting two input files:

File01: Hello World Goodbye World

File02: Hello Hadoop Goodbye Hadoop

This example aims to calculate the quantity of each word. Firstly, the input file is read,

the data is delivered to Map program. After the Map program reads the input, it cut out simple words and marks it for 1, and then transform the original form into <word, 1> form. Secondly, the data is delivered to reduce program, collecting the same key of value, and thus taking shape <word, list of 1> form. Finally, the added 1 value is the number of word. Figure 2.1 shows a data flow diagram of 'WordCount'.

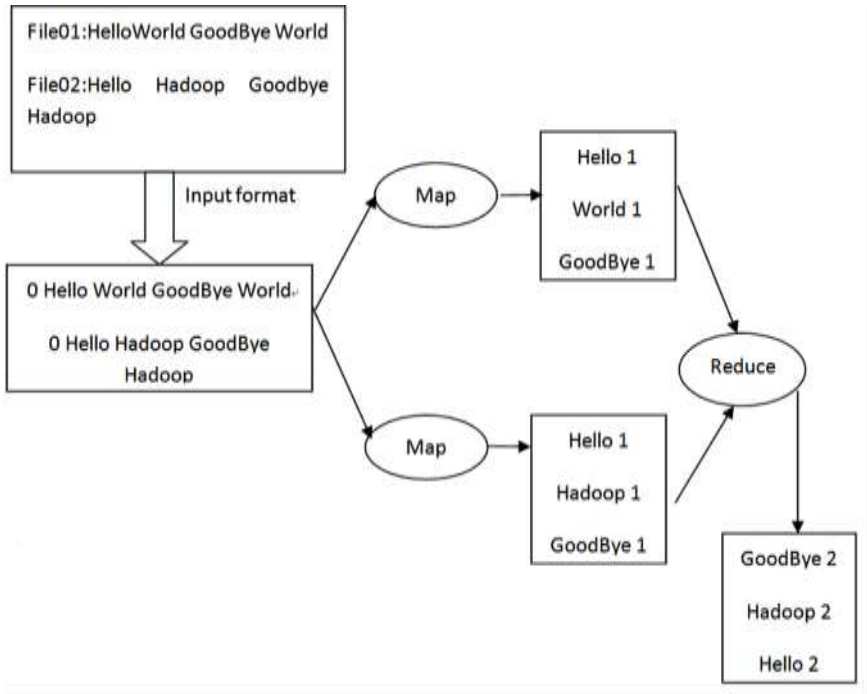


Figure 2.1: Data flow diagram of WordCount.

2.2.2 MapReduce Execution Flow

In MapReduce, the input file will be divided into several blocks by Map function. Assuming that the file is divided into M blocks, Map function will start M tasks. The results which include the value sets which have the same key will be sent to Reduce task [25]. Thus, N Reduce tasks start to process the results in parallel. MapReduce execution process is described as follow [26]:

- MapReduce system will divide the input file into blocks, and then parallelly execute JobTracker and TaskTracker on Hadoop cluster. JobTracker is the main program responsible for distributing task to TaskTrackers dynamically according to the load of each node in cluster. TaskTracker will process both

Map and Reduce task.

- Executing Map tasks. Map task will analyse a set of key-value pairs from the local file block. After the Map function finishes processing, it outputs a key-value pair of intermediate result which will be stored in the memory of each node.
- The key-value pair in memory will eventually be written to local disk. According to the law of the function, different key values will be distributed into R sections. Meanwhile, each position information of intermediate result of Map task and partition information will be sent to the JobTracker.
- Executing Reduce task. It will merge the intermediate results with the same key. According to the sections, the key-value pairs will be sent to relevant Reduce task. Finally, Reduce task outputs R files which are storing the results of Reduce task.
- In the end, JobTracker notifies the user to return calling point.

An example of MapReduce program execution flow is shown in Figure 2.2.

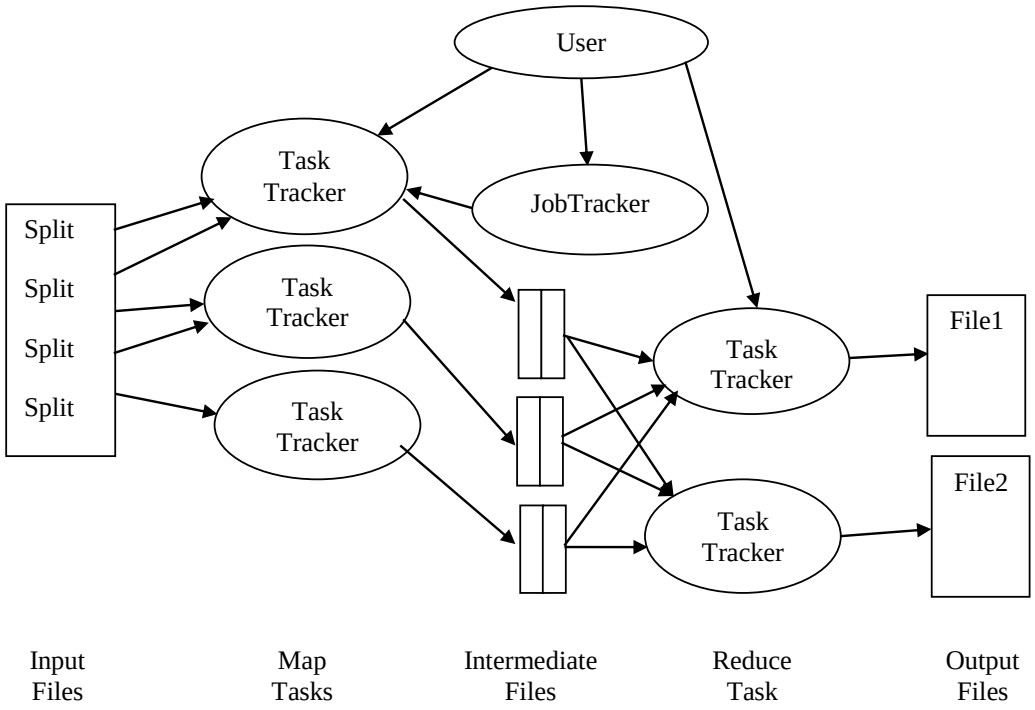


Figure 2.2: MapReduce program execution flow.

2.2.3 MapReduce Fault-tolerant Mechanism

In the real environment, sometimes malfunction happens to the machine, e.g. the code fails or process crashes. The following will focus on analysing error handling mechanism.

- *Hardware failure*

From the perspective of MapReduce task execution, the hardware failure includes JobTracker and TaskTracker.

JobTracker failure is a single point failure, which is the most serious error. Up to now, the only way to avoid the failure is to create multiple spare JobTracker nodes. After the main JobTracker fails, the Leader election algorithm is used to choose a new JobTracker node. By contrast, the TaskTracker failure happens more than JobTracker. When the TaskTracker fails or runs slowly, TaskTracker will inefficiently send the heartbeat to JobTracker or even stop. If the JobTracker and the TaskTracker do not communicate for a specific time period, JobTracker will return to this task immediately. If this TaskTracker is still at the map stage, JobTracker will command another TaskTracker to re-perform this task; if this TaskTracker is at the Reduce stage, the JobTracker will command another TaskTracker to continue the Reduce task.

- *Task failure*

In practical task, MapReduce will encounter user code fault or processing crashes resulting in task failing. The fault code leads it to throw an exception during execution, and JVM process will automatically exit. Meanwhile, it will send an error message to TaskTracker and the error message is written to the log file. In the end, the TaskTracker will mark the task attempt fails. For the task failure caused by processing crashes, listener will find the process exit. At the moment, the task will be also marked as failed attempts by TaskTracker. Eventually, TaskTracker will apply to JobTracker for a new task.

2.3 HDFS Mechanism

HDFS (Hadoop Distributed File System) is a distributed file system designed to run on commodity hardware [27]. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. The following description will be divided into several aspects.

2.3.1 HDFS Features and Limitations

HDFS has the following features:

- *Processing large files*

The large files mean the size of files over hundreds MB, even over hundreds TB.

- *Streaming Data Access*

HDFS is designed based on 'write once, read multiple' task. It means one data set generated by the data source, will be copied and distributed to different nodes, then response to a variety of data analysis requests.

- *Running on a cluster of cheap commercial machines*

Hadoop is designed to run on low requirement hardware. Therefore, it can run on inexpensive commodity hardware cluster.

Because of the above considerations, HDFS has some main limitations when processing certain issues including:

- *Unsuitable for low latency data access*

HDFS is not suitable for processing some low delay application requests. It is specifically for processing big data analysis with high latency to generate great data throughput.

- *Unable to store a large number of small files efficiently*

Hadoop needs NameNode to manage the metadata of file system to response to the requests from the client returning file location. Therefore, the limit of the quantity of

files is determined by NameNode.

- *Unable to support multi-user write and freely modify files*

There is only one writer in a file in HDFS. Meanwhile, writing operation can only be completed at the end of the file, which is able to execute additional operations.

Although there are some existing problems, with the researchers' efforts, HDFS will be more advanced and satisfy more application requirements in the future.

2.3.2 HDFS Related Concepts

This section introduces some concepts related to HDFS.

- *Data Blocks*

HDFS block is an abstract concept, whose default size is 64MB. It is same with single file system, the file on HDFS is also divided into blocks for storage and it is a logical unit of a file storing process. Abstract block will bring lots of benefits. Firstly, arbitrary size of files could be stored Hadoop distributed file system. For example, it is impossible to store a 100 TB data for a single node. However, in Hadoop distributed file system, the file will be divided into many blocks and stored in the different machines in the cluster. Secondly, to simplify the storage subsystem, using abstract block as a unit of operating system. The size of block in Hadoop Distributed File System is fixed, thus facilitating the storage system managing especially for the metadata and file block can be stored separately. In addition, the blocks were used more conveniently to generate data copy from fault-tolerant in distributed system. In order to handle node failures in HDFS, the default file block will be set in three copies and stored on different nodes of the cluster. When a block is damaged, the system will get metadata information from NameNode, then read and store a copy on another machine [28].

- *NameNode and DataNode*

There are two types of nodes in HDFS architecture. One is responsible for Master task that is NameNode, the other one is responsible for Slave task that is DataNode.

NameNode manages file system namespace, maintains the file directory tree of file system and the index directory. However, these is not permanently stored information, the NameNode will rebuild information dynamically when the system starts. DataNode is a worker node in file system to perform specific task: storing file blocks and called by client and NameNode. Meanwhile, it will regularly send the information of file block to NameNode by heartbeat.

2.3.3 HDFS Framework

As shown in Figure 2.3, HDFS uses Master/Slave framework for file system management. An HDFS cluster consists of a NameNode and a number of DataNodes. NameNode is a central server responsible for managing namespace of file system and accessing from client. DataNode generally runs a single node process responsible for managing storage on this node. Internally, a file is divided into one or more databases that are stored on a set of data nodes. NameNode executes the namespace operation of file system, such as open, close, rename or directory, decides the mapping from data block to specific nodes. DataNode is responsible for processing the read/write requests from file system client. Creating, deleting and copying the data block under the control of the NameNode.

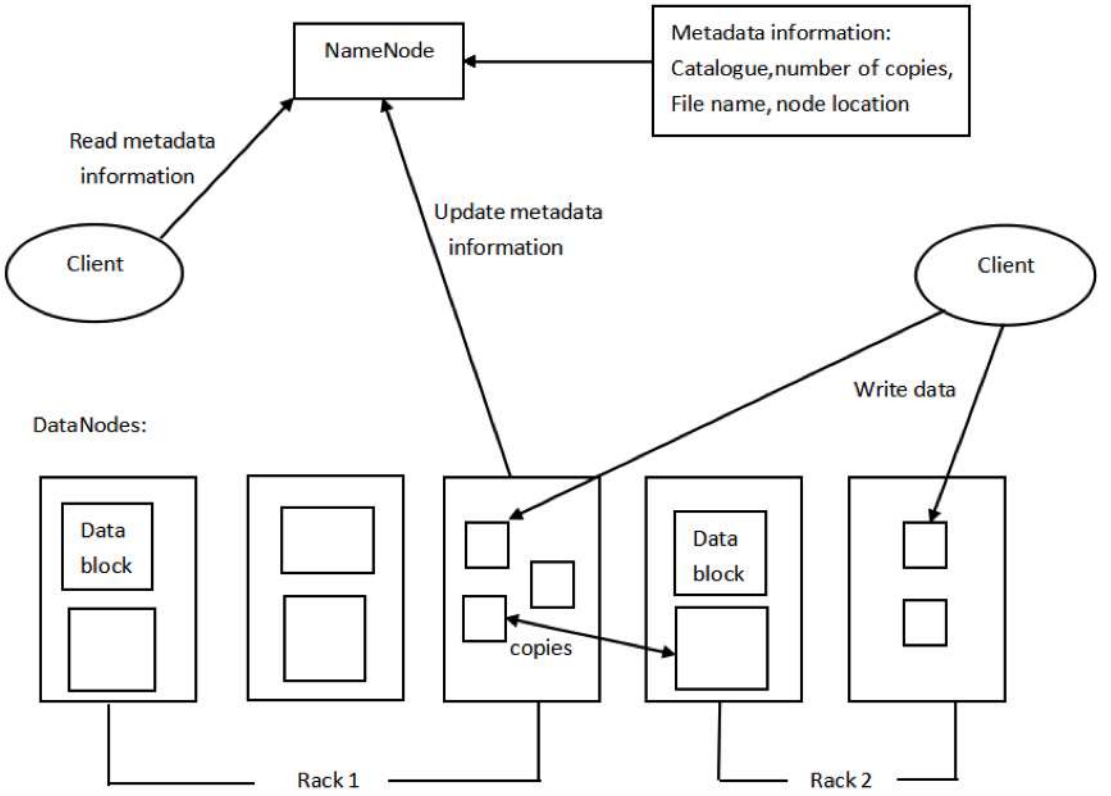


Figure 2.3: HDFS internal framework.

- *Storing and reading a copy of the policy*

In HDFS, the key of reliability and performance is copy storage, optimizing the strategy of copy storage is the significant feature different from most of the other distributed file systems. HDFS uses rack-aware strategy to improve the data availability, reliability and network bandwidth utilization. Large HDFS instances are generally run on the clusters which are composed of computers from several racks. The computers in the different racks require communication through the switches. In most cases, the bandwidth of the two computers in the same rack is larger than the bandwidth of the two computers in the different racks.

On the other hand, HDFS tries to make the program read the nearest copy of the client in order to reduce the overall bandwidth consumption and reduce bandwidth-delay when it reads data. If there is a copy on the same rack, it will be read directly. If the HDFS cluster spans multiple data centres, the client will be the first attempt to read the copy from the local data centre.

- *Safe mode*

After the NameNode start, it enters in a special state called safe mode, but it will not copy the data block. NameNode receives heartbeats and block status reports from all DataNodes. Block status report includes all the data blocks from the DataNode. Each data block has to specify the minimum number of copies. When NameNode testing confirms the copies of a data block reaching a minimum value, the data block will be considered as secure. After a certain percentage of the data block is detected in secure, NameNode will exit from safe mode state. After that, it will determine the data blocks did not reach the specified number, then copy these data blocks to the other DataNode.

- *File security*

The importance of NameNode is outstanding; the client is not able to get the location of the file blocks without it. In practice, if the NameNode of the cluster fails, all the files in the entire file system will be lost, because the file blocks on DataNode cannot be used to reconstruct the file.

The first method to ensure the safety of NameNode in Hadoop is the backup of the metadata files which are persistently stored on NameNode, then dump to another file system which is synchronous operation. The usual implementation approach is dumping the metadata from Namenode to remote NFS file system. The second method is running on secondary NameNode in the system synchronously. The main role of this node is to periodically merge and edit the namespace mirror in the log in order to avoid editing log excessively. Running secondary NameNode requires a lot of CPU and memory to do the merge operation, so it needs to run on a separate machine. The merged namespace mirror will be stored on the secondary NameNode and will be used to make the bench after the NameNode crashed, so that the loss of files is minimized.

2.4 Mahout Algorithm Library

2.4.1 Introduction

Apache Mahout originates from 2008, when it was an Apache Lucene subproject. Apache Lucene is a well-known open source search engine, which implements advanced information retrieval, text mining functions. In the field of computer science, these concepts are similar with machine learning [29]. Therefore, the initial Apache Mahout was composed of machine learning algorithms. In a short time, Apache Mahout absorbed an open-source collaborative filtering algorithm project called 'Taste'. After two years of development, Apache Mahout eventually became the top project of Apache. Apache Mahout aims to build a scalable machine learning algorithm library and the scalability for big data sets. Mahout algorithm runs based on Hadoop platform and realizes by the use of MapReduce Model. However, Mahout is also available on a single node or other non-Hadoop platforms, the non-distributed algorithms in Mahout core library also have perfect performance.

Apache Mahout is a new open-source project from Apache Software Foundation (ASF), which supports a variety of classical machine learning algorithm, in order to help researchers to develop more intelligent applications easily and quickly. Although plentiful techniques and algorithms have been achieved, there are some new algorithms being developed and tested. Mahout project currently includes the following four parts:

Clustering: Dividing the input data into multiple classes by similar objects.

Classification: Using the existing classification file classifier to classify the unclassified files.

Recommendation engine (collaborative filtering): Discovering the users' behaviours and recommending the items that customs might be loved.

Mining frequent item sets: Using an item set (check records or shopping directory) to

identify projects appeared at the same time.

2.4.2 Taste

Taste is the efficient implementation of a personalized recommendation engine which is Apache Mahout provided. The engine is based on java implementation with high scalability, while some recommendation algorithms in Mahout performed MapReduce programming model transformation. Therefore, it can use Hadoop distributed architecture to improve the performance of recommendation algorithm.

Taste is not only suitable for java applications, and also acting as a component of the internal server to provide the logic of recommendation using HTTP and Web Service. The Taste is designed so that it can meet the high performance, flexibility and scalability requirements. Figure 2.4 shows the core components of the Taste.

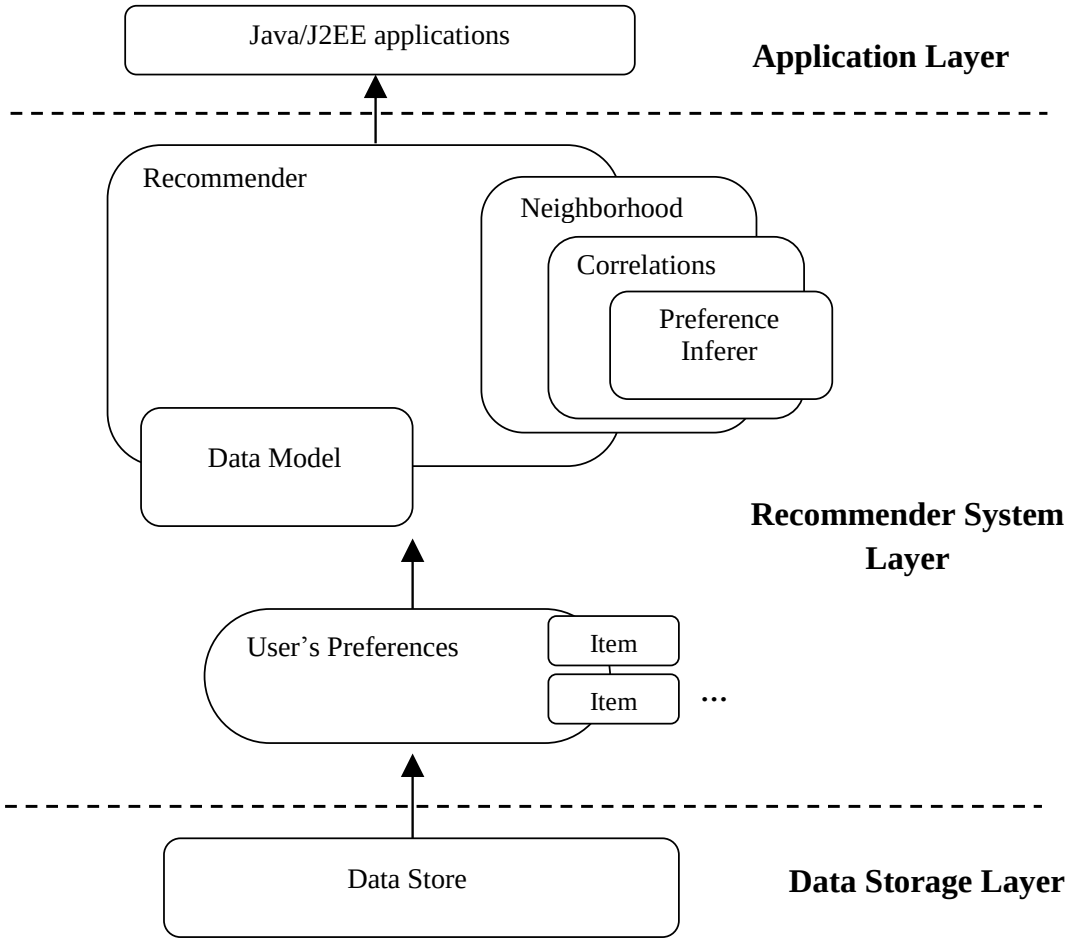


Figure 2.4: Core components of Taste.

2.4.3 Mahout Algorithm

Table 2.1 shows parallelization strategy algorithm implemented in Mahout.

Table 2.1 Mahout algorithm library

Category	Name
Clustering	Canopy Clustering
	K-means Clustering
	Fuzzy K-means
	Streaming K-means
	Spectral Clustering
Classification	Logistic Regression
	Naive Bayes/Complementary Naive Bayes
	Random Forest
	Hidden Markov Models
	Multilayer Perceptron
Collaborative Filtering	User-Based Collaborative Filtering
	Item-Based Collaborative Filtering
	Matrix Factorization with ALS
	Matrix Factorization with ALS on Implicit Feedback
	Weighted Matrix Factorization, SVD++
Dimensionality Reduction	Singular Value Decomposition
	Stochastic SVD
	PCA(via Stochastic SVD)
	QR Decomposition

Miscellaneous	RowSimilarityJob
	ConcatMatrices
	Collocations
	Sparse TF-IDF Vectors from Text
	XML Parsing
	Email Archive Parsing
	Lucene Integration
	Evolutionary Processes
Topic Models	Latent Dirichlet Allocation

Mahout transfers lots of algorithms which are previously used on the single machine to MapReduce mode by parallelization strategy. The advantage is to use these algorithms in Hadoop platform for machine learning and data mining tasks, in order to greatly improve the capability for processing big scale data sets.

2.5 Summary

This chapter introduces an overview of Hadoop framework, analyses the MapReduce programming model and the implementation mechanism of HDFS distributed file system. Moreover, it briefly introduces Mahout algorithm library which is one of Hadoop subprojects.

Chapter 3

The Parallel Strategy of K-means Algorithm

3.1 Clustering Analysis

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters) [30]. Clustering is an unsupervised learning to divide some unmarked objects into several meaningful clusters. Moreover, in a cluster, it is unknown how many the target data sets exist classes, using distance metric for clustering all the objects, so that the distance in the same cluster is minimal as well as the distance in the different clusters is maximal. There are some main clustering algorithms including [31]:

- *Division method*

For a given data set consisting of N tuples or records, the Division method will construct K clusters (K≤N), and K clusters meet the following conditions: Firstly, each cluster contains at least one data object. Secondly, each data record only belongs to one cluster. For the given parameter K, the algorithm gives an initial division method, and then changes divisions by iteration ensuring that the improved division is better than the previous one. The typical algorithms are K-means algorithm and K-medoids algorithm.

- *Hierarchy method*

This method hierarchically decomposes the given data set until meeting certain condition. It includes the ‘bottom-up’ condensation method and the ‘top-down’ secession method. The typical algorithms contain Birch algorithm, Cure algorithm, Chameleon algorithm, et al.

- *Density-based method*

This method based on a variety of distance has fundamental difference with the other methods. In this method, as long as there is a density of point in the area larger than a certain threshold, it will be put in the similar clustering algorithm. The typical algorithms contain DBSCAN algorithm, OPTICS algorithm, et al.

- *Grid-based algorithm*

In this method, the first step is dividing the data space into limited number of grids, and in the meantime, each unit is regarded as a single object in all of the processing. The prominent advantage of the method is high processing rate, and normally it is independent from the number of records in target data set, but related to the number of units divided from data space. The typical algorithms are STING algorithm, Wave-Cluster algorithm, et al.

- *Model-based algorithm*

Model-based method is assuming a model for each cluster, and then finding data for the given model for optimal simulation. The given model probably is the density distribution function of data point in space. Generally, it includes statistics scheme and neural network scheme.

K-means algorithm is one of classical clustering analysis algorithms. The next two sections will focus on parallel strategy analysis and research of K-means algorithm based on Mahout framework.

3.2 K-means Algorithm Descriptions

K-means clustering is a method of vector quantization, originally from signal processing, which is popular for cluster analysis in data mining. K-means clustering aims to partitioning observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells [32]. According to the input parameter K, K-means divides N object sets into K clusters, in order to make the high

similarity within clusters and low similarity between the clusters. The similarity of cluster is the mean measurement of the objects in the cluster. K-means algorithm processes as follow: Firstly, it randomly selects K objects, and each object represents an initial cluster centre [33]. For each of the remaining objects, according to their distance from the clusters centre, it will be assigned to the most similar cluster. Afterwards, the new centre of each cluster is calculated. This process is repeated until the criterion function convergence happens. The Formula (3.1) shows the criterion function.

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} | p - m_i |^2 \tag{3.1}$$

In the formula, p is the certain point in space, demonstrating a given object; m_i is the mean value of cluster C_i (p and m_i are multidimensional); E indicates the sum of square error of all of the objects in the cluster. Namely, for each object in each cluster, calculating the square of its distance from the cluster centre, and thus summing the square values. K-means algorithm process is described as follow [34]:

(1) Establishing the initial division and obtaining K clusters according to the given K.

(2) Calculating the distance from each point to the cluster centre, adding it to the nearest cluster.

(3) Recalculating the centre of each cluster.

(4) Repeating process (2) and (3), until the centre of each cluster is maintained in an accuracy range or has reached the maximum number of iterations.

There is an example of which they are nine points and K is 3 to illustrate K-means algorithm. Before clustering, nine points in a two-dimensional plane were randomly selected, i.e. coordinates (7,8), (12,1), (13,6), (13,13), (13,19), (14,5), (17,16), (19,20), (20,7).

- *The first clustering*

System randomly selects three points as cluster centre, assuming the points are (7, 8), (12, 1), (13, 6). Next, it will calculate the distance from each point to each cluster

centre; the point will belong to the nearest cluster centre. After calculation, point (7, 8), point (13, 19) are in cluster A, point (12, 1) is in cluster B, point (13, 6), point(13, 13), point(14, 5), point(17, 16), point(19, 20), point(20, 7) are in cluster C. The Figure 3.1 shows the results.

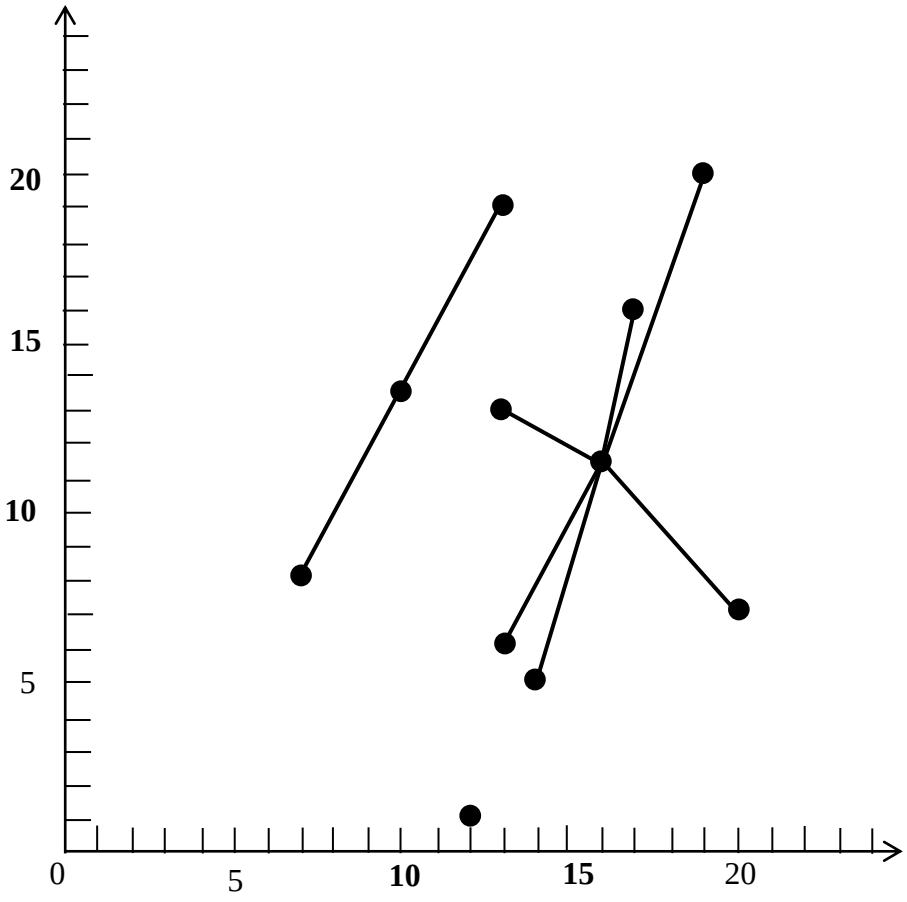


Figure 3.1: Result of first clustering.

When the cluster centre has been updated, the new cluster centre value is the average of all members of the cluster. The new cluster centre of cluster A is (10.0, 13.5), the new cluster centre of cluster B is (12.0, 1.0), the new cluster centre of cluster C is (16.0, 11.2).

- *The second clustering*

According to the previous generation of the cluster centre, the distance from each point to each centre is recalculated and clustering is repeated. The new results are point (7, 8), point (13, 13), point (13, 19) constitute new cluster A, point (12, 1), point

(13, 6), point (14, 5) constitute new cluster B, point (17, 16), point (19, 20), point (20, 7) constitute new cluster C. The new cluster centre of cluster A is point (11.0, 13.3), the new cluster centre of cluster B is point (13.0, 4.0), and new cluster centre of cluster C is point (18.7, 14.3). Figure 3.2 shows the result.

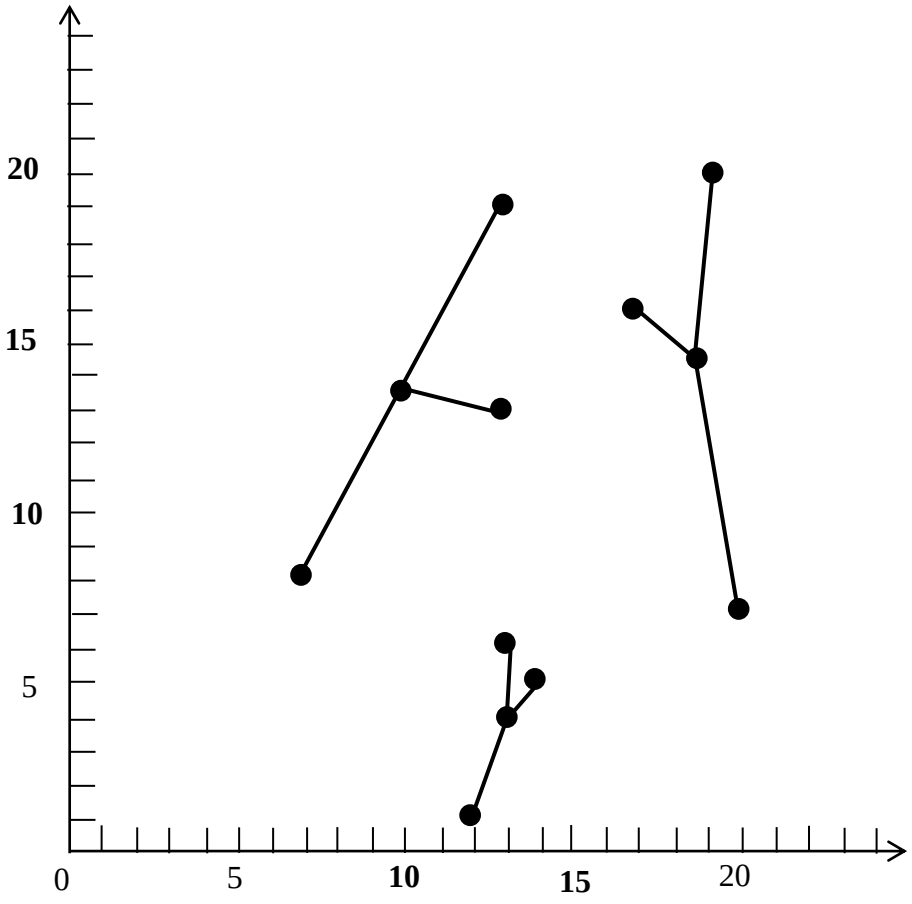


Figure 3.2: Result of second clustering

After the clustering operation had been repeated, the clustering results were found no change and were able to meet the convergence conditions. Thus, the clustering finished.

The above example illustrates that K-means algorithm has numerous advantages. The principle of K-means algorithm is relatively easy to be achieved with high efficiency of execution and scalability for big data. However, the disadvantages of K-means algorithm are also obvious. Firstly, it requires the user to confirm the number of clusters before executing cluster, but they usually find an optimal value of K through

several experiments. Secondly, due to the algorithm randomly selects the initial cluster centres at the beginning, the algorithm has poor tolerance of noise and isolated points. Noise is the incorrect data in the cluster; isolated point is the data which is far away from other data points with low similarity. At the beginning, if the isolated point or noise is selected as the cluster centre, there will be a problem during the clustering process. The way to solve the problem and optimize the execution process is to quickly choose the K value and initial cluster centre.

3.3 K-means Algorithm Parallel Strategy

This section presents the design strategy in parallelization of K-means. It first introduces Mahout data model.

3.3.1 Mahout Data Model

Mahout data model is discussed from the aspects of vector representation, similarity calculation, converting data to vector and converting text to vector.

3.3.1.1 Vector Representation

The object of Mahout clustering algorithm is represented as the simple data model: Vector. On the basis of the vector data description, it easily calculates the similarity of two objects. Vector is a complex object that each domain is a floating-point number (double). Some of the values of vector are very dense, each domain has its value; some of the values of vector are sparse, there are probably only some domains having their values. Due to the different data contents of vector in specific applications, Mahout supports plenty of implementations [35].

- *DenseVector.*

For DenseVector, the implementation is a floating-point array. All domains in vector will be stored, suitable for storing dense vectors. This vector representation is highly efficient when most of the vectors have their valid value. It allows quick access to the

value of vector in any dimension and sequentially to traverse all dimensions of vectors.

- *RandomAccessSparseVector*

The implementation is HashMap based on floating-point. The key is int type and value is double type. The value of dimension will be stored when it is not zero. When some dimensions of a vector are nonzero, using RandomAccessSparseVector to present vector have higher memory efficiency than using DenseVector, but the speed of accessing dimension value and sequentially traversing all dimensions of vectors are slower.

- *SequentialAccessSparseVector*

The implementation is a parallel array, int type and floating-type separately presents the dimension and dimension values of the stored vector. It also only stores the nonzero value of vector. Thus, it combines the advantages of the previous two vectors, but the disadvantages are the poor speed of random inserting and selecting.

Based on the above three vectors, Mahout algorithm is able to be achieved by following the data characteristics and data access methods. If the algorithm for vector value has plentiful random inserting and updating, it should choose DenseVector or RandomAccessSparseVector to present vector; if most of them are sequential access, SequentialAccessSparseVector would show better performance.

3.3.1.2 Similarity Calculation

Clustering is a data object divided into numerous clusters, which aims to make the objects in the same cluster with high similarity as well as the objects in the different clusters with high distinction. Mahout algorithm represents the object as a simple model called vector. Actually, similarity calculation is to calculate the distance between two vectors, the smaller distance means higher similarity. There are some calculation distance methods shown as follow:

- *Euclidean Distance*

Euclidean Distance is one of the most simple distance measuring methods. It measures the absolute distance of each point in an n-dimensional space. Assuming x, y are two points in an n-dimensional space, the Formula (3.2) shows the Euclidean Distance.

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{3.2}$$

Mahout uses a class called 'EuclideanDistanceMeasure' to calculate the distance.

- *Squared Euclidean Distance*

This value of distance is the square of return value of Euclidean Distance. Formula (3.3) shows the distance n-dimensional space.

$$d(x,y)=\sum_{i=1}^{n}(x_i-y_i)^2 \tag{3.3}$$

Mahout uses a class called 'SquaredEuclideanDistanceMeasure' to calculate the distance.

- *Manhattan Distance*

Manhattan Distance is from the distance among city districts. Unlike Euclidean Distance, it is the summation of the distance among the multiple dimensions. Figure 3.3 indicates that the distance of two points in XY plane calculated through Manhattan Distance and Euclidean Distance.

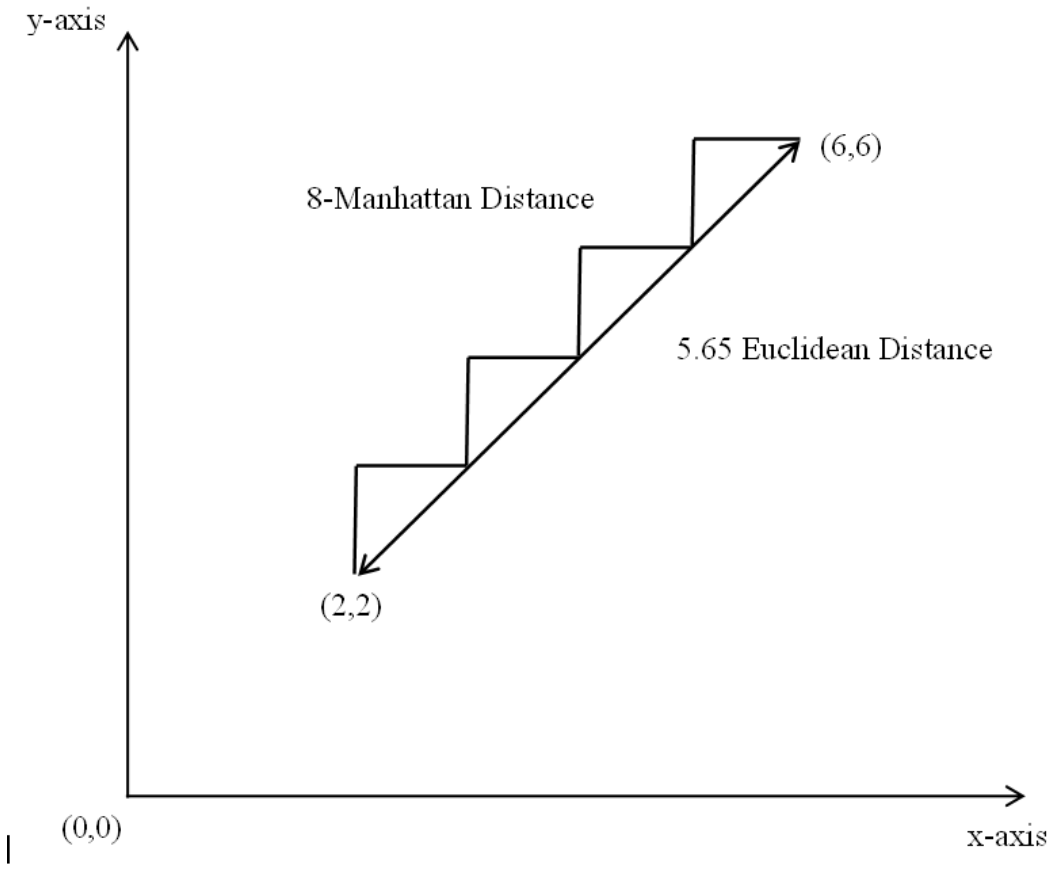


Figure 3.3: Comparison between Euclidean Distance and Manhattan Distance.

In Figure 3.3, the result of Manhattan Distance is 8 and the result of Euclidean Distance is 5.65 between point (2, 2) and point (6, 6). Mahout uses a class called 'ManhattanDistanceMeasure' to calculate the distance. The Formula (3.4) shows the Manhattan Distance.

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i| \tag{3.4}$$

- *Cosine Distance*

Cosine Distance regards the points as vectors, that there is a small angle between two vectors. It calculates the cosine of the angle, the bigger value of cosine means the smaller angle, and the smaller value of cosine means the bigger angle. The Cosine Distance formula (3.5) of two n-dimensional vectors is shown.

$$d(x,y)=\cos\theta=\frac{x*y}{\|x\|\|y\|}=\frac{\sum_{i=1}^{n}x_iy_i}{\sqrt{\sum_{i=1}^{n}x^2}*\sqrt{\sum_{i=1}^{n}y_i^2}} \tag{3.5}$$

Mahout uses a class called ‘CosineDistanceMeasure’ to calculate the distance.

- *Tanimoto Distance*

In some applications, it considers not only the vector angle problem, but also the useful information contained on the length of the vector. For example, assuming there are three vectors(X(20,20), Y(40,40), Z(50,50))in plane space. Although any two vectors cosine is 0, vector Y and vector Z are closer. Cosine Distance only considers the angle and ignores the length; Euclidean Distance only considers the length and ignores the angle. Therefore, in this condition both of them are invalid. Tanimoto Distance is also regarded as generalized Jaccard Distance, simultaneously reflects the impact of angle and distance. The Tanimoto Distance formula (3.6) is shown.

$$d(x,y)=\frac{x*y}{\|x\|+\|y\|-xy}=\frac{\sum_{i=1}^{n}x_i y_i}{\sum_{i=1}^{n}x_i^{\,2}+\sum_{i=1}^{n}y_i^{2}-\sum_{i=1}^{n}x_i y_i} \tag{3.6}$$

- *Weighted Distance*

Mahout provides WeightedDistanceMeasure class using Euclidean Distance and Manhattan Distance. Because of the importance of different dimensions of a vector are different, the different dimensions need to be distinguished. According to the requirements, Weighted Distance will give specific weighting factors to specific dimension, in order to influence the measurement results. For example, when calculating the distance between two points on a plane, the X-direction and Y-direction can be imparted weighting factor 1 and 2. The value of Y-direction obviously has greater effect on distance.

3.3.1.3 Converting Data to Vector

For object clustering, firstly it needs to be converted to a vector. The dimension vector size will depend on how many features the object has.

For example, if there is an orange dataset, the feature includes the weight, colour and size. Firstly, converting the features to dimensions, assuming weight is 0, colour is 1

and size is 2. Secondly, quantize each dimension. For the weight, it is assigned grams or kilograms to measure; for size, it is assigned centimetres for the diameter of oranges to measure. For colour, the best method to represent it is using wavelength of colours. It will be shown the different colours accurately. Table 3.1 described some oranges vectors.

Table 3.1: Orange vectors.

Orange	Weight(kg) (0)	Colour (1)	Size (2)	Vector
Small circle green orange	0.11	510	5	[0.11,510,5]
Big ellipse red orange	0.23	650	10	[0.23,650,10]
Small thin long red orange	0.09	630	4	[0.09,630,4]
Big circle yellow orange	0.25	590	12	[0.25,590,12]
Medium ellipse green orange	0.18	520	8	[0.18,520,8]

Here are the data sets of oranges vectorization code:

```
1    public static List<Vector>creatOrangeVector(){
2        List<Vector>oranges= new ArrayList<Vector>();
3    NamedVector orange=new NamedVector(new DenseVector(new double[]
{0.11,510,5}),);
4    oranges.add(orange);
5        orange=new NamedVector(new DenseVector(new double[]{0.23,650,10}),);
6        oranges.add(orange);
7        orange=new NamedVector(new DenseVector(new double[]{0.09,630,4}),);
8        orange.add(orange);
9        orange=new NamedVector(new DenseVector(new double[]{0.25,590,12}),);
10       orange.add(orange);
11       orange=new NamedVector(new DenseVector(new double[]{0.18,520,8}),);
12       orange.add(orange);
13       return oranges;
}
```

3.3.1.4 Converting Text to Vector

With the development of the information age, the number of text files increase rapidly. Generally, the useful information should be extracted from massive text data. During the text data pre-processing, it is a vital step to convert high-dimensional text data to vector. Vector Space Model (VSM) is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings. It is the most common similarity calculation model. Its basic concept is converting text to vector by calculating the similarity to compare texts [36].

Assuming there are M files, File D_i= (T_{i1}, W_{i1}, T_{i2}, W_{i2}..., T_{in}, W_{in}), where T_{ij} is a lexical item, W_{ij} is the weight of this lexical item. The degree of correlation between file D_i and file D_j commonly measured using the similarity. The equations shown as follow.

$$sim(D_i, D_j) = \cos\theta = \frac{\sum_{k=1}^{n} W_{ik} \bullet W_{jk}}{\sqrt{\sum_{k=1}^{n} W_{ik}^2} \bullet \sqrt{\sum_{k=1}^{n} W_{jk}^2}} \tag{3.7}$$

For example, assuming there are ten words: T_1, T_2..., T_{10} in two articles: D_1, D_2. The number of words after statistics is listed in Table 3.2.

Table 3.2: The number of words.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
D_1	1	2		5		6	7		9	
D_2		3		4			6	8		10

This table is the vector space model, for any two files, it can be selected to calculate the cosine of the two vectors when it wants to calculate their similarity. Assuming in the file D_1, there are ten words and each word appears a certain amount of times, represented as a_1, a_2..., a_{10}; and in the file D_2, there are ten words and each word is appeared b_1,b_2...,b_{10}. Then the cosine value of D_1 and D_2 can be shown as follow:

$$\cos\theta = \frac{d_1^T d_2}{\| d_1 \| \| d_2 \|} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_{10} b_{10}}{\sqrt{a_1^2 + a_2^2 \cdots a_{10}^2} \bullet \sqrt{b_1^2 + b_2^2 + \cdots b_{10}^2}}$$
$$= \frac{2\times 3 + 5\times 4 + 7\times 6}{\sqrt{1^2 + 2^2 + \cdots + 9^2} \bullet \sqrt{3^2 + 4^2 + \cdots + 10^2}} = \frac{68}{14\times 15} \approx 0.324 \tag{3.8}$$

If cosine value is 1, then the two files are completely same. If cosine value is 0, then the two files are completely different. Overall, the increasing cosine value in [0, 1] results in higher similarity of two files. After calculation, the cosine of two articles is approximately 0.324.

There are some simple processing methods counting the keywords to calculate the

similarity of files. However, when the number of a keyword analogously occurs in two articles which have a greater gap of length, it will reduce the accuracy of the results. For example, if a word is appeared 100 times in both file D_1 and file D_2, it has the same importance from the view of word frequency. However, if it considers another factor that the keyword of file D_1 is 1,000 and the keyword of file D_2 is 10,000, the importance of this keyword is different in file D_1 and file D_2. Therefore, the occurrence frequency of a word in a file is regarded as the weight of the word, is called term frequency (TF). This method is to process word-counting, using keywords divided by the total number of the keywords in the file. Therefore, for the word T_i in a particular file, the word frequency is shown.

$$TF_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{3.9}$$

Where, n_{ij} is the number of the word appearing in the file D_j, while the denominator is the sum of all words in the file n_{ij}.

If it only distinguishes the file using word frequency, it will encounter another problem. There are lots of common words such as you, me in an article with high occurrence. Regardless of using which kind of distance to measure the similarity of two articles, these words usually give some negative impact to the result. TF-IDF (Term Frequency-Inverse Document Frequency) technology can be used to sort out this problem. Actually, TF-IDF is TF*IDF, TF is word frequency representing the frequency of words in the document and IDF is inverse document frequency whose main idea is that if the document contains fewer words, the IDF will be bigger, it indicates that this word has a good ability to distinguish categories [37].

TF-IDF is a statistical method for evaluating the importance of a word or phrase in a file in a file set or corpus. The importance of this word or phrase proportionally increases with the increasing number of appearances in the file, but it inversely proportional decreases with the frequency of appearances in the corpus. The main idea of TF-IDF is that if the TF of a word or phrase in an article is higher and rarely appears in the other articles, then it is believed that the word or phrase has a good

ability to distinguish categories and is suitable for classification [38]. Actually, TF-IDF means TF*IDF, that TF means term frequency which is the frequency of the word in a file and IDF means inverse document frequency, that if the number of files contain fewer word 't', then IDF is higher. It can use the number of total files divided by the number of files which contain the specific word, and then the calculated logarithmic quotient is the IDF of the particular word [39].

$$IDF(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{3.10}$$

In the formula, N is total number of documents in the corpus, $|\{d \in D : t \in d\}|$ means number of document when the term t appears.

For example, there is a file includes 100 words and 'Mahout' appeared three times, and then the frequency of 'Mahout' appeared in the file is 3/100=0.03. If 'Mahout' is appeared in 1,000 files, and the total number of the files is 10,000,000. Its inverse file frequency is log (10,000,000/1,000) =4. Finally, the score of TF-IDF is 0.03*4=0.12.

The high frequency word in a specific file as well as the word in the entire corpus with low frequency will generate a high weight of TF-IDF. Thus, TF-IDF tends to filter out common words and retain the significant words.

Vector space model (VSM) is based on assuming that the words characteristics are completely independent with each other. For example, in a two-dimensional model, x-coordinate is completely independent from the y-coordinate. However, in practice it is not very realistic, such as 'cola' is the most likely associated with 'coco', these two words are not independent. The other models attempt to consider the independence of the words; Latent Semantic Indexing (LSI) is one of well-known techniques. LSI integrates the words into a whole set to detect dimension, but until now, Mahout has not yet implemented this feature. Nevertheless, the TF-IDF method based on the assumption of independence has achieved a great result. Mahout uses a method called 'n-grams' to solve the problem of words independence.

N-gram is an ordered permutation of words, one word is called unigram and two

words are considered as an integral called bigram, such as ‘coca cola’. Three or more words are called trigrams, 4-grams, 5-grams et al. Classical TF-IDF method assumes that a word is independent from the other words. Created vectors using this method generally lack the ability to discriminate the key feature of files. To solve the problem, Mahout implements a high technology which identifies the words appear synchronously with high probability, such as ‘Martin Luther King’ or ‘Coca Cola’. The vectors’ dimension will be mapped to bigrams rather than unigrams. In a sentence with multiple words, it is able to construct the n-grams by choosing consecutive N blocks. This operation can generate many n-grams, but most of them do not have practical significance. Such as ‘Welcome to Apache Hadoop’, it will generate some bigrams shown as follow:

‘Welcome to’

‘to Apache’

‘Apache Hadoop’

Some phrases (‘Welcome to’, ‘Apache Hadoop’) have favourable feature used to generate file vectors, but some phrases (‘to Apache’) do not have. If it wants to evaluate the importance of unigrams and bigrams in a file, firstly it needs to connect and use the TF-IDF weighting technique. If meaningless bigrams are calculated, a greater amount of inverse document frequency will be generated, and will be given a higher weight. However, the TF-IDF believes the greater frequency of bigrams is more useless, and then it will be filtered out. For this problem, Mahout uses log-likelihood technology which is able to determine whether two words are occurring occasionally and simultaneously or are from a meaningful unit, and to choose the meaningful words and discard the meaningless words. Using this method in TF-IDF weighting, some more meaningful bigrams will be more appropriately declared, like ‘coca cola’. In Mahout, TF-IDF weighting technology and n-gram technology are implemented in DictionaryVectorizer class, in order to convert the text to vectors.

3.3.2 K-means Parallel Design Based on MapReduce

The data sets processing by MapReduce model have some characteristics: it can be divided into several small data sets and each data set can be processed completely in parallel. In the K-means algorithm, calculating the distance from each point to cluster centre is processed independently. There are no interaction among the points, thus, it can use the MapReduce model to achieve parallelism. Figure 3.4 shows the process of one iteration [40].

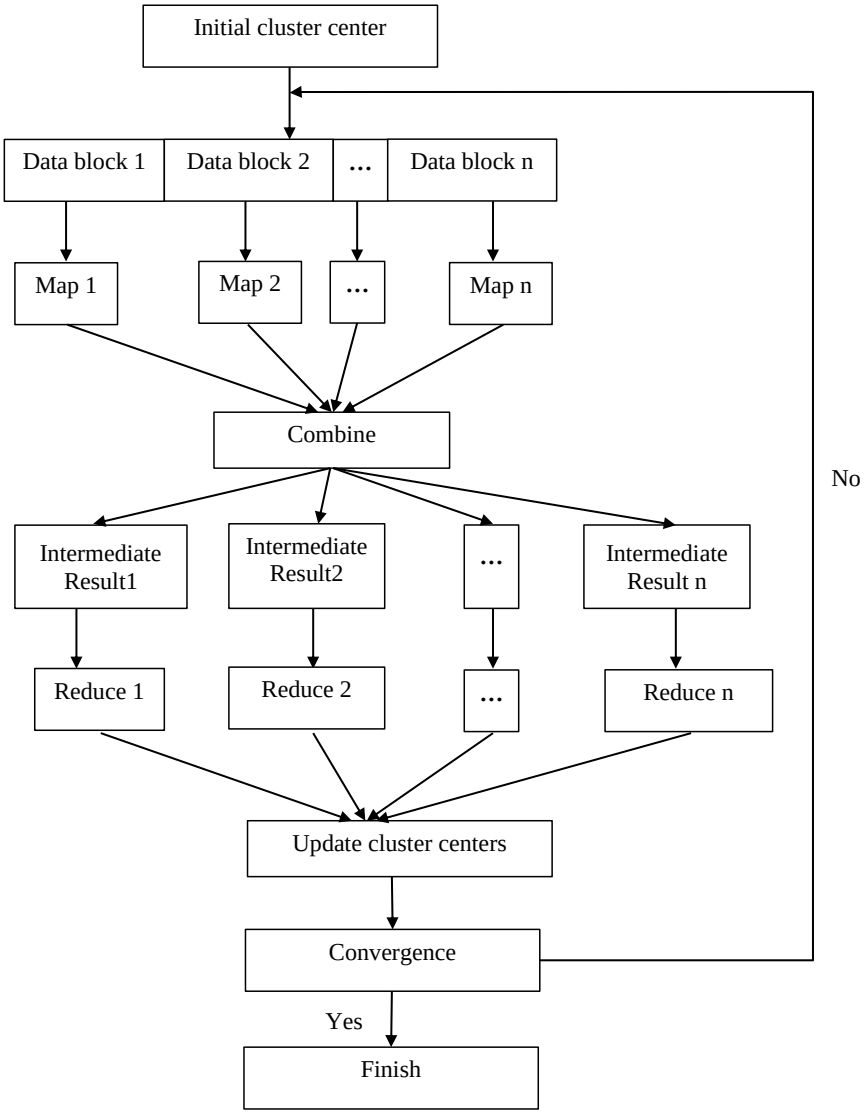


Figure 3.4: Flowchart of one iteration.

3.3.2.1 Feasibility Analysis of Parallel Algorithm

There are two sections when K-means algorithm doing iteration: The first one is calculating the distance between each data object and all cluster centres, which is the most time-consuming part of the algorithm; The second one is generating new cluster centres that calculating the average of all data objects in each cluster. With the increasing size of the data, the time of calculating the distance between object and cluster centre is increasing. Therefore, in order to improve the efficiency of clustering, the K-means algorithm parallelization was employed to solve the problem.

In the MapReduce programming model, a large data is divided into many file blocks, distributedly stored in computer clusters. The K cluster centres are stored in each cluster node, calculating the distance of a data object to all cluster centres on one node. Meanwhile, the other nodes can calculate the distance between the other data objects and all cluster centres with multiple Map tasks processing in parallel. After all the data objects allocation are completed, the data belonging to the same cluster will be sent to a cluster node, the data belonging to the different clusters will be sent to the different nodes. Thus, the new clusters in parallel on K clusters can be updated, which are parallel multiple Reduce tasks.

The serial implementation of K-means algorithm does not only requiring large memory, but the time complexity is also relatively high. Its executing time is n*k*t*o, n means the number of data, k means the number of clusters, t means the number of iterations, o means the time calculating the distance from data point to the cluster centre in one iteration and sending the data point. Assuming there are 10,000 data objects and K is 100. In one iteration, it will complete 1,000,000 times distance calculation from each data object to each cluster centre. This process is too time-consuming, but after parallel process, the efficiency of this part will be greatly enhanced. If there are 10,000 Map tasks, while a data object is calculating the distance from it to 100 cluster centres, the other data objects are calculating the distances at the same time, which is assigned a Map task for each object.

3.3.2.2 Parallel Algorithm Design

In the K-means algorithm parallelization in each iteration corresponding to a Job, that is a MapReduce operation: calculating the distance of each data object to all cluster centres and sending the data object corresponding to a Map task, updating the cluster centre corresponding to a Reduce task. The data records of dataset are stored in rows, in order to expediently start Map task. Therefore, each Map task automatically gets a record for executing and is finished automatically by MapReduce framework.

- *Design of Map function*

The task of Map function is calculating the distance from data object to cluster centres and sending the data object to the nearest cluster. The input data format is <key,value>, key is the number of row, value is the data record. In addition, each Map function also reads K cluster centres, which are initial cluster centres or the cluster centres after several iterations. The format of Map function output is <key,value>, key is clustering ID, value is data record. The pseudo code of Map function is shown as follow:

```
1   void map (Writable key, Text point) {
2       min_distance=MAXDISTANCE;
3           for(i=0;i<k;i++){
4               if(distance(point,cluster[i]<min_distance){
5               min_distance=distance(point,cluster[i]);
6               currentCluster_ID=I;
7           }
8       }
9   Emit_Intermediate(currentCluster_ID,point);
10  }
```

In this pseudo code, the 'for' circulation is traversing k points, if there is a distance

less than min_distance, and then re-assign the min_distance and get serial number of the cluster. The line 8 is classifying the cluster according to the points and corresponding serial number.

- *Design of Reduce function*

After obtaining the intermediate results from Map function, Reduce function will update the corresponding centre of the cluster ID and output the new centre of each cluster. These data records set are composed of all the intermediate results from the Map task. The pseudo code of Reduce function is shown as follow:

```
1    void reduce(Writable key, Iterator<PointWritable>points){
2        num=0;
3        While(points.hasNext(){
4            PointWritablecurrentPoint=points.next();
5            num+=currentPoint.getNum();
6            for(i=0;i<dimension;i++)
7            Sum[i]+=currentPoint.point[i];
8        }
9        for(i=0;i<dimension;i++)
10       mean[i]=sum[i]/num;
11       Emit(key,mean);
12   }
```

In this pseudo code, the 'while' circulation is traversing each point, and then calculating the summation of each dimension of each point in clusters. Finally, it will respectively calculate the mean of each dimension and determine the new cluster centre that is the mean.

When the Reduce tasks are completed, the change of the centre will be calculated to

see whether it is within a threshold range. If it is in the threshold range, the final result will be output; otherwise, it will start a new MapReduce task following the K new centres to iterate again. Before getting the final clustering results, it needs to start another Job only includes Map task to allocate all data points.

3.3.2.3 The Time Complexity of Algorithm

The datasets are stored as blocks on the distributed cluster nodes in which default size on HDFS is 64MB. According to the number of blocks, MapReduce will start corresponding number of Map tasks. The Map tasks execute in parallel on multiple cluster nodes. Therefore, the most time-consuming part of K-means algorithm is assigned to the clusters to execute in parallel. Assuming start p Map tasks, the time complexity of K-means algorithm is n*k*t*o/p.

3.4 Summary

This chapter discussed the clustering classification, Mahout data model and vectorization of the text files. Then studied the parallel design of K-means algorithm based on MapReduce, and analysed the parallel feasibility of algorithm, implementation of parallel and parallel time complexity. Due to calculate the distance from each point to the cluster centre independently and it is the most time-consuming part of K-means, it is able to parallelize the distance calculation part in the K-means.

Chapter 4

Experimentation and Evaluation

This chapter aims to analyse and evaluate the impacts of the Hadoop parameters and compare the efficiency between parallel and serial K-means algorithm through experiments.

4.1 Building experimental platform

Due to the limited experimental conditions, there is only one computer in the experimental platform while Oracle Virtualbox is used to set up four virtual machines. It includes one Namenode and four Datanode. The environment configuration is shown in Table 4.1.

Table 4.1: Environmental configuration.

Node Type	CPU	Memory	Hard Disk
Master	8 Core	8GB	1TB
Slave1	1 Core	1GB	128GB
Slave2	1 Core	1GB	128GB
Slave3	1 Core	1GB	128GB
Slave4	1 Core	1GB	128GB

4.1.1 Software Environment

Operation System: Ubuntu 12.04

JDK Version: JDK 1.6

Hadoop Version: Hadoop-1.2.1

Mahout Version: Mahout-0.6

4.1.2 Set up Hadoop Platform

Cluster consists of five nodes, including one Namenode and four Datanode. The nodes are connected by a network, node IP address is assigned as follow:

Namenode *134.83.181.43*

Slave1 *134.83.181.89*

Slave2 *134.83.181.91*

Slave3 *134.83.181.93*

Slave4 *134.83.181.83*

After configuring the Hadoop environment, it is able to check the platform working status and nodes status from *http://localhost:50030*. The status parameter is shown in Figure 4.1 and Figure 4.2.

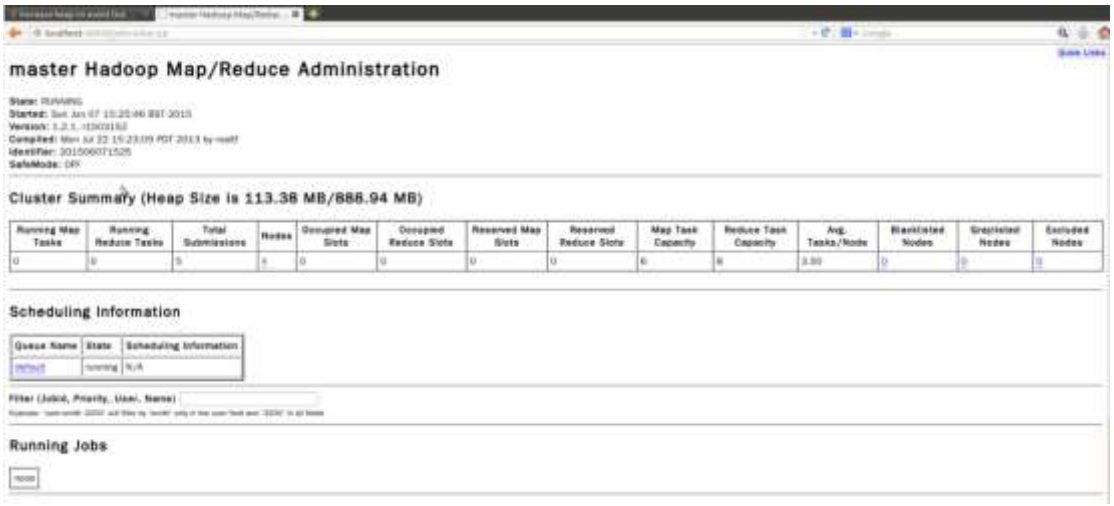
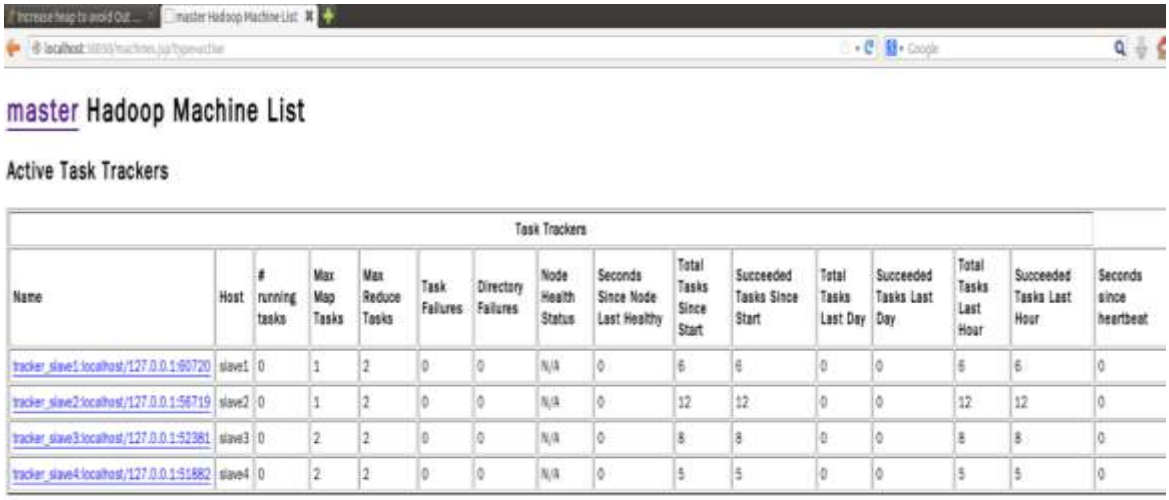


Figure 4.1: Hadoop platform working status.



master Hadoop Machine List

Active Task Trackers

Task Trackers

Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day	Total Tasks Last Hour	Succeeded Tasks Last Hour	Seconds since heartbeat
tracker_slave1:localhost/127.0.0.1:60720	slave1	0	1	2	0	0	N/A	0	6	6	0	0	6	6	0
tracker_slave2:localhost/127.0.0.1:56719	slave2	0	1	2	0	0	N/A	0	12	12	0	0	12	12	0
tracker_slave3:localhost/127.0.0.1:52381	slave3	0	2	2	0	0	N/A	0	8	8	0	0	8	8	0
tracker_slave4:localhost/127.0.0.1:51882	slave4	0	2	2	0	0	N/A	0	5	5	0	0	5	5	0

Figure 4.2: Nodes working status.

4.2 Experimental Studies

4.2.1 Efficiency of K-means Algorithm in Serial and Parallel Environment

Executing the K-means algorithm in Mahout contains three main steps which are shown as follow:

(1) Using seqdirectory command to convert the original files into sequence files [41].

(2) Using seq2sparse command to convert the sequence files into vectorial files.

(3) Using K-means command to execute K-means clustering algorithm.

The data used in experiments was 'Bag of words' dataset. The dataset records the words from different texts and is suitable for clustering analysis. The total data is around 7.3 GB. In the experiment, 10 million to 300 million records to run five groups will be extracted. Table 4.2 shows the number of records and size of records in each data.

Table 4.2: Data information.

	Number of records (million)	Size of records (MB)
A	10	80.57
B	20	162.13
C	50	403.73
D	200	1612.33
E	300	2413.58

4.2.1.1 Pre-processing Data

In order to improve operation efficiency, Mahout uses the seqdirectory command to convert the '.txt' file into sequence files which actually calls the SequenceFileFromDirectory class. The command is shown.

```
bin/mahoutseqdirectory -c UTF-8 -i /testdata -o /testdata-seqfiles (1)
```

In the command, -c option specifies the encoding of the input file; -i option indicates the pending text input path; -o option represents the output sequential file path. The next job is uploading the processed data in the HDFS, the command is shown.

```
bin/hadoop fs -put /home/hadoop/testdata-seqfilestestdata-seqfiles (2)
```

In order to make clustering algorithm easier to use, Mahout also need to use seq2sparse command to convert sequence file into vectorial files. This command calls SparseVectorFromSequenceFiles class, the command is shown.

```
bin/mahout seq2sparse -i /testdata-seqfiles -o /testdata-vectors -wt TFIDF -ng 2 (3)
```

The input path is testdata-seqfiles which saves sequence files. The output path is testdata-vectors. In the command, -wt indicates the weighting method, which is

usually TF or TFIDF; -ng represents the maximum value of N-gram that is assigned 2 meaning generating unigrams and bigrams, the default value is 1; 1-norm represents the Manhattan distance, 2-norm represents the Euclidean distance. Using the above command will generate two directory files and five folders in the output folder which are shown in figure 4.3.

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
df-count	dir				2015-05-20 16:59	rwxr-xr-x	hadoop	supergroup
dictionary.file-0	file	6.25 MB	2	64 MB	2015-05-20 16:57	rw-r--r--	hadoop	supergroup
frequency.file-0	file	5.02 MB	2	64 MB	2015-05-20 16:59	rw-r--r--	hadoop	supergroup
tf-vectors	dir				2015-05-20 16:59	rwxr-xr-x	hadoop	supergroup
tfidf-vectors	dir				2015-05-20 17:00	rwxr-xr-x	hadoop	supergroup
tokenized-documents	dir				2015-05-20 16:55	rwxr-xr-x	hadoop	supergroup
wordcount	dir				2015-05-20 16:57	rwxr-xr-x	hadoop	supergroup

Figure 4.3: Output Folder.

df-count catalogue: Saving the frequency information of texts.

dictionary.file-0 catalogue: Saving the glossary of these texts.

frequency.file-0 catalogue: Saving the glossary corresponding frequency information

tf-vectors catalogue: Saving the text vector which is regarding TF as weight.

tfidf-vectors catalogue: Saving the text vector which is regarding TFIDF as weight.

tokenized-documents catalogue: Saving the text information after partition.

wordcount catalogue: Saving the number of all words.

4.2.1.2 Executing K-means Process

From the directory structure, it needs two input directories: one for saving text vector, the other for saving initial partition of the point. Afterwards, using the K-means command to run K-means clustering algorithm, the command is shown.

```
bin/mahoutkmeans -i /testdata-vectors/tfidf-vectors -c /euclidean
-dmorg.apache.mahout.common.distance.EuclideandistanceMeasure -o /result -x 200
-k 20 -ow --clustering                                                    (4)
```

In the command, the input path is /testdata-vectors/tfidf-vectors and the output path is /result; -k represents the number of clusters; -c represents the folder with the initial cluster centres, after the K value is specified, it will randomly selects K vectors to write to the directory; -dm option determines the distance measurement method, which focuses on checking the similarity between the cluster centre and each point; -x represents the maximum number of iterations of clustering, 200. K-means is an iterative algorithm, therefore it changes this option to ensure the algorithm calculated completely; -ow is used to indicate whether to overwrite the output folder so that the Mahout is able to prevent the unfinished output data from being damaged by newly generated data. After running the above command, the implementation of the work will be shown at http://localhost:50030, as Figure 4.4 shows.

Figure 4.4: Result Demonstration.

The first job is Kmeans Driver randomly divides the initial data which is written into the path:/euclidean.

The second job is Kmeans Driver initiates to build the first iteration of clusters. The input path is /testdata-vectors/tfidf-verctors and the initial partition path is /euclidean, the output path is /result/clusters-1.

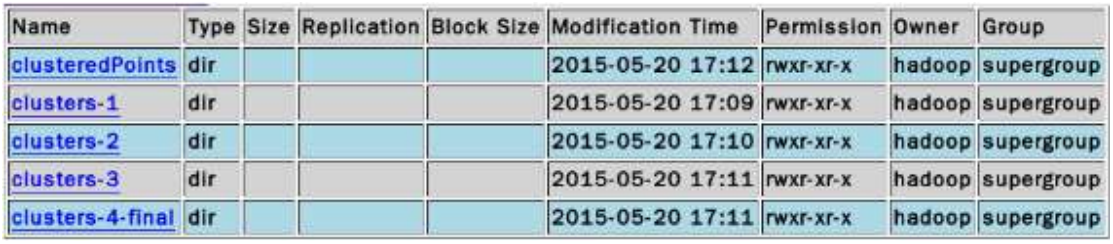
The third job is Kmeans Driver initiates to build the second iteration of clusters. The input path is /result/cluster-1 and the output path is /result/cluster-2.

The fourth job is Kmeans Driver initiates to build the third iteration of clusters. The input path is /result/cluster-2 and the output path is /result/cluster-3.

The fifth job is Kmeans Driver initiates to build the fourth iteration of clusters. The

input path is /result/cluster-3 while there are two output paths which are /result/clusters-4-final and /result/clusteredPoints.

After execution, the result folder will be shown at http://localhost:50070 or it is checked on Hadoop distributed file system. Figure 4.5 shows the result.



Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
clusteredPoints	dir				2015-05-20 17:12	rwxr-xr-x	hadoop	supergroup
clusters-1	dir				2015-05-20 17:09	rwxr-xr-x	hadoop	supergroup
clusters-2	dir				2015-05-20 17:10	rwxr-xr-x	hadoop	supergroup
clusters-3	dir				2015-05-20 17:11	rwxr-xr-x	hadoop	supergroup
clusters-4-final	dir				2015-05-20 17:11	rwxr-xr-x	hadoop	supergroup

Figure 4.5: Result folder.

During the stage of generating clusters, each iteration will generate a path, and the output path of the previous iteration is regarded as the input path of the next iteration. This kind of directory is named 'Cluster-N', and the final result of clustering will be saved in the fold which is named 'Cluster-total number of iterations-final'. The final result of clustering points is saved in the fold named 'clusteredPoints', the cluster-id and documents-id will be shown.

In the experiment, the software of serial K-means clustering algorithm is 'Weka', which is an open-source data mining tool developed by University of Waikato, New Zealand [42]. It includes most of the data mining algorithms and is widely used in academia.

The serial experiment uses the Simplekmeans algorithm based on Weka platform, and then compared the time efficiency with running K-means algorithm on Hadoop platform. The configuration parameters of serial experiment test are the same as those on parallel environment. Figure 4.6 shows the experimental environment.

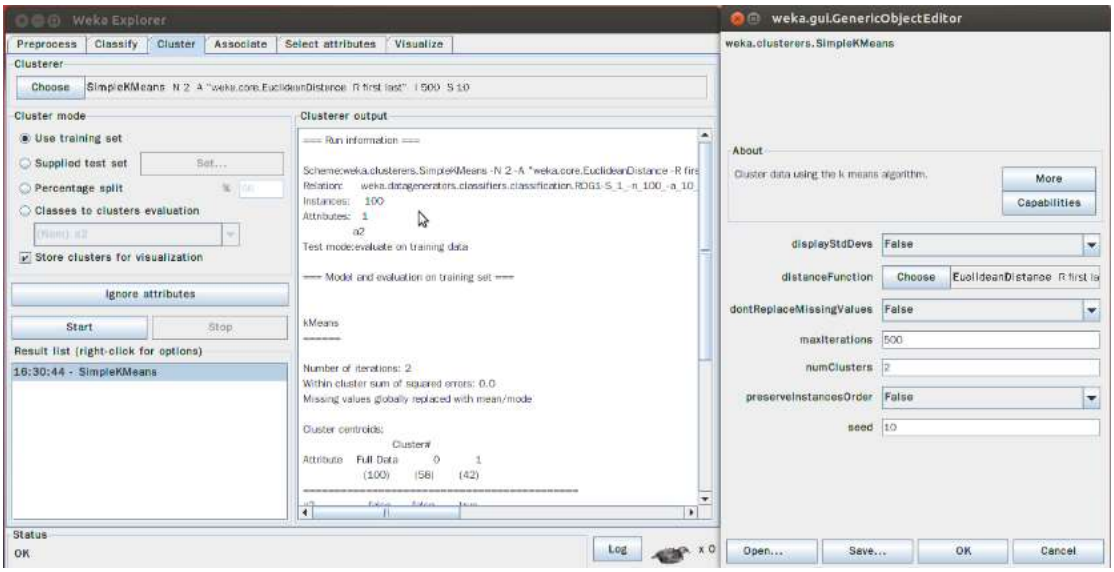


Figure 4.6: Weka working Status.

4.2.1.3 The Analysis of Experiment

The experiment compares the time efficiency of K-means clustering algorithm between serial implementation and parallel sing-node implementation. In the same software and hardware environment, with increasing data, the time from reading data to completing K-means clustering process is calculated. Firstly, five groups of data are imported into Weka as well as execute K-means clustering algorithm, and then execution time is calculated. Secondly, five groups of data are imported into Hadoop distributed file system platform, successive increasing Datanode from one and executing K-means clustering algorithm respectively and calculating execution time simultaneously. All the parameters of K-means are the same for Weka platform and Hadoop platform. Table 4.3 shows the experimental results.

Table 4.3: Experimental results.

	1Datanode(s)	2Datanode(s)	3Datanode(s)	4Datanode(s)	Serial(s)
A	218.1	232.8	220.8	212.6	153.7
B	388.2	326.0	308.7	298.5	313.5
C	642.7	568.5	502.1	473.8	869.2
D	1933.2	1678.5	1567.8	1425.1	Out of memory
E	2751.3	2213.8	1938.5	1756.1	Out of memory

Firstly, in the parallel environment, the K-means clustering algorithm of five datasets with the increasing number of nodes from one to four is executed respectively, and then is compared with the execution time. Figure 4.7 shows the result.

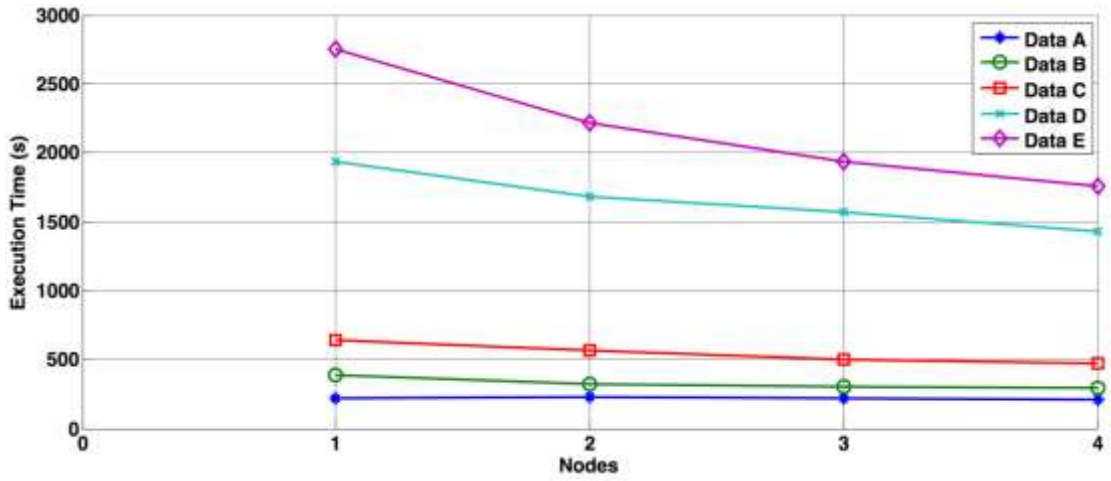


Figure 4.7: Parallel environment.

The result demonstrates that due to small data size, the system cannot fully take advantage of the map function, and the execution time in different number of nodes is approximate. However, for the same size of data, in the case of increasing the node, the system will decrease the time to complete the task. Thus, increasing the number of

nodes can significantly improve the system capabilities for processing the same scale data.

Moreover, five groups data execution time between serial environment and single-node parallel environment is compared, Figure 4.8 shows the result.

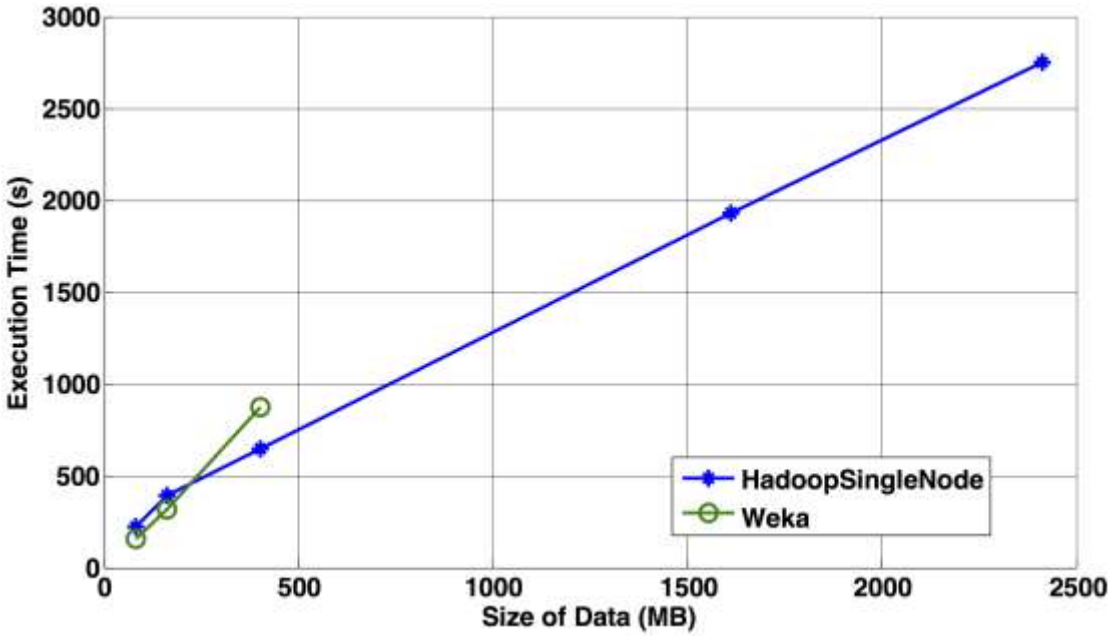


Figure 4.8: Efficiency between single node and serial.

The result shows that when the data is small size, the execution time of serial K-means clustering algorithm is lower than execution time of parallel K-means clustering algorithm. Each iteration of K-means clustering algorithm will restart a new job, and starting and interactive process will consume some resources. Iterative MapReduce task is dividing the task into one Map task and one Reduce task. In addition, each task is independent, the Hadoop platform will continually read, write and transfer data. When the dataset is tiny, the proportion of practical executing time in the whole time is ignorable. However, when the amount of data increasing to a certain degree, the time spend by serial clustering algorithm is much higher than it spend by parallel algorithm. Because the system resource is unable to afford the overhead of the serial computing, so that it possibly will be thrown out of memory exception. Nevertheless, even there is only one node in Hadoop platform, it is able to easily deal with the increasing dataset, and successfully complete the computing task

based on large dataset. Thus, the K-means clustering algorithm with MapReduce distributed implementation which is based on Hadoop distributed file system has good reliability, fully reflecting the advantages of Hadoop to process large amounts of data. Mahout platform is based on Hadoop distributed file system, so that changing Hadoop parameters can improve the performance of Mahout. The next section will introduce the optimization of Hadoop parameters.

4.2.2 Optimization of Hadoop Parameter

There are more than 190 parameters in Hadoop configurations, which control application execution process. Users can adjust the value of these parameters according to their requirement [43]. A large number of experiments illustrate that the parameters have significant impact on some aspects of Hadoop, such as memory allocation and usage rate, parallel degree, I/O optimization, network bandwidth usage rate, etc. The values of different parameters constitute a huge parameter configuration space. Table 4.4 presents the Hadoop core parameters.

Table 4.4: Parameters of Hadoop.

Configuration Parameters	Default Values	Data Types	Descriptions
io.sort.factor	10	integer	The number of streams that can be merged while sorting.
io.sort.mb	100	integer	The size of the in-memory buffer assigned to each task.
io.sort.spill.percent	0.8	float	A threshold which determines when to start the spill process, transferring the in-memory data into the hard disk.
mapred.reduce.tasks	1	integer	The number of reduce task(s) configured for a Hadoop job.
mapreduce.tasktracker.map.tasks.maximum	2	integer	The number of map slots configured on each worker node.
mapreduce.tasktracker.reduce.tasks.maximum	2	integer	The number of reduce slots configured on each worker node.
mapred.child.java.opts	200	integer	The maximum size of the

			physical memory of JVM for each task
mapreduce.reduce.shuffle. Input.buffer.percent	0.70	float	The amount of memory in percentage assigned to a reducer to store map results during the shuffle process.
mapred.reduce.parallel. Copies	5	integer	The number of parallel data transfers running in the reduce phase.
mapred.compress.map. Output	False	boolean	Compression of map task outputs.
mapred.output.compress	False	boolean	Compression of reduce task outputs.

io.sort.factor: After a map task has been executed, there are several stream files on local disk. This parameter determines the number of files will be merged simultaneously in the last step of map task. The default value is 10, but the speed of sorting process is independent on the value, therefore we need appropriate adjustments based on the data.

io.sort.mb: When a map task start operating and generating intermediate data, the intermediate data is written into an in-memory buffer instead of it is directly written into hard disk. When the buffer reaches a certain threshold, it will start a background thread to sort the contents of buffer, and then written to the local disk as a spill file. The default value of this parameter is 100M, for the huge cluster can be set to 200M. The recommended value for this parameter is between 30% and 40% of Java_opts value.

io.sort.spill.percent: This value is above buffer threshold which default value is 80%. When the data were saved in the buffer reaching the threshold, the background thread will sort the data in the buffer, and then write into the hard disk. The value of this parameter is preferably not less than 50%.

io.sort.reduce.tasks: The default value of this parameter is 1 and the selection of appropriate value can significantly influence the performance of a Hadoop job. The optimal value for this parameter mainly depends on the size of the input data and the available reduce slots configured in a Hadoop cluster. When the input data is small, a

small parameter value decreases the overhead in setting up tasks and a large number of reduce tasks is preferred on large input data due to the improvement on the hard disk IO utilization. Generally, the recommended number of reduce tasks is set to ninety per cent of the total number of reduce slots configured in a Hadoop cluster.

mapreduce.tasktracker.map.tasks.maximum,

mapreduce.tasktrakcer.reduce.tasks.maximum: These parameters indicate that the number of map and reduce tasks in each tasktracker can be ran simultaneously, the default value is 2. The value is depending on the physical memory of each node. The recommended value for these parameters is the number of CPU cores,-1.

mapred.child.java.opts: This parameter manages the size of memory of sub-process which is started by task. The default value is -Xmx200m that the memory of sub-process of task is up to 200MB. In general, increasing the value of this parameter can effectively improve the performance of MapReduce.

mapreduce.reduce.shuffle.input.buffer.percent: This parameter configures that the consumption memory of the data is read from map task account for heap memory; the default value is 0.7(70%). Due to the maximum heap memory of reduce task is set to 1GB, so that by default, when the usage space of heap memory achieves 700MB, it will write the results into the HDFS.

mapred.reduce.parallel.copies: This parameter assigns the number of parallel transmission of reduce during the copy phase. The default value is 5, but for the large scale clusters can be adjusted to 20-50.

mapred.compress.map.out, mapred.output.compress: These two parameters are responsible for whether the map output and reduce output is compressed, the data type is Boolean and the default value is false. If it is configured to true, the map output and reduce output will be compressed. Compression will consume more CPU resources, but can reduce the transmission time. Otherwise, it will take up more bandwidth. In order words, this is CPU exchange for IO. In general, CPU is not the performance bottleneck, so configures the parameter to true is very beneficial.

The Hadoop clusters configuration environment is same as the previous environment. The data is from 'Bag of words' dataset which records abundant word from texts. The number of records is 2,000 million and the size is 1612.13MB. During the experimental procedure, the WordCount application is run as a Hadoop job to analyse the impacts of configuration parameters on Hadoop performance. By modifying the value of io.sort.factor, number of reduce tasks and Java_opts parameters, the same data is input separately, and then the execution time is recorded. The Figure 4.9 shows the impacts for Hadoop by modifying io.sort.factor.

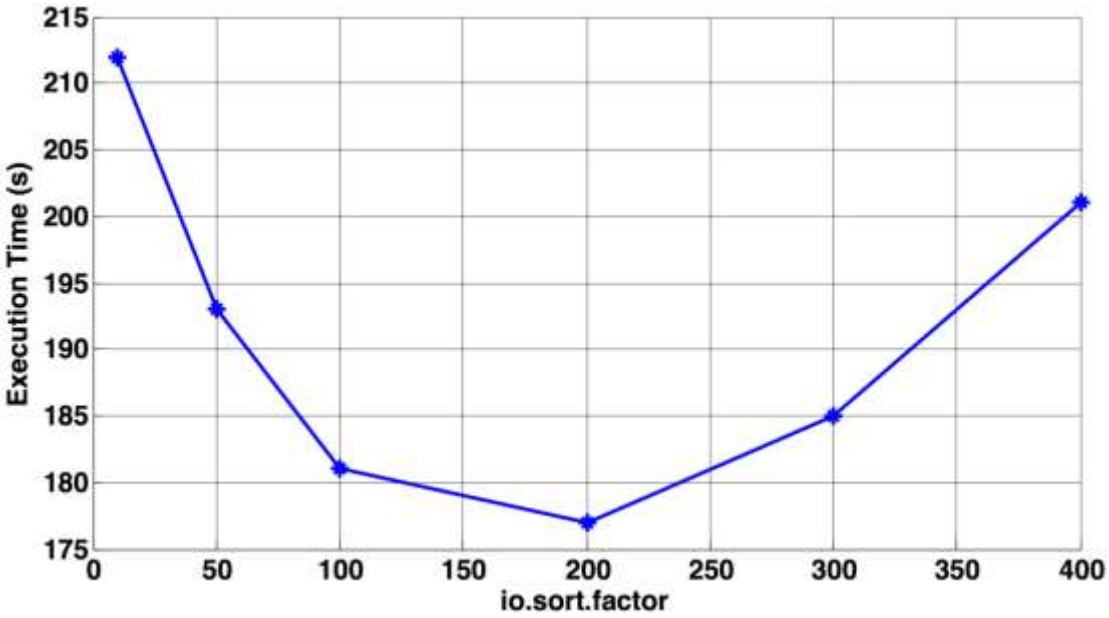


Figure 4.9: The impacts of io.sort.factor.

The io.sort.factor represents the number of data streams which are able to be merged in the sorting process. With the io.sort.factor increasing until 200, which is the best value of the parameter, the execution time goes down. Subsequently, the execution time goes up with io.sort.factor increasing further. This is due to the trade-off between the added overhead incurred in merging the data streams and the reduced overhead incurred in IO operations when io.sort.factor increases.

The Figure 4.10 shows the impacts for Hadoop by modifying the number of reduce tasks.

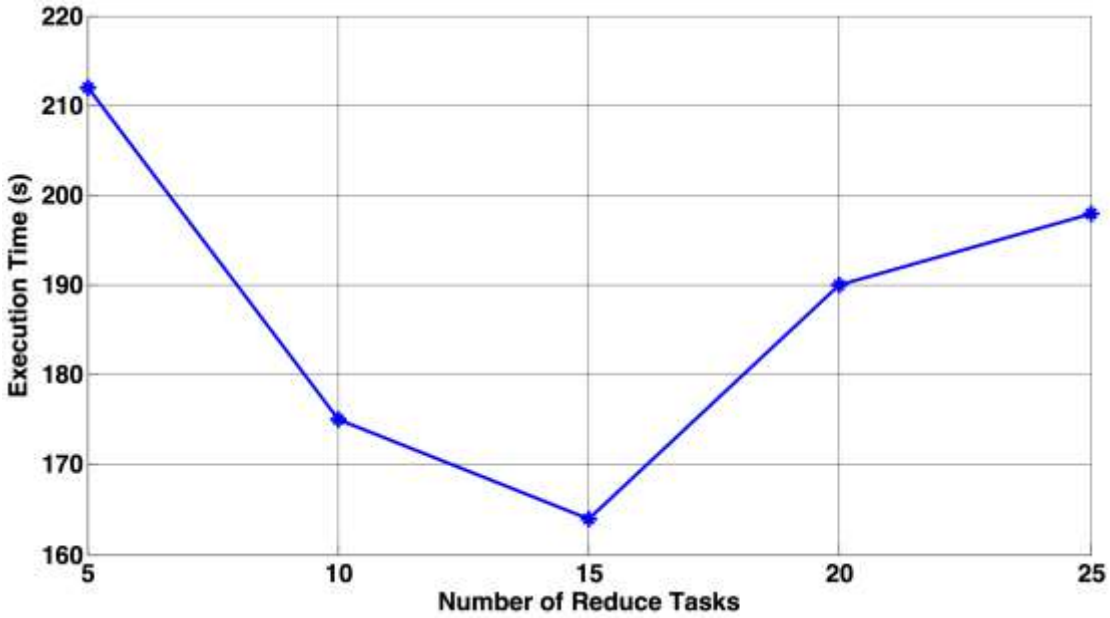


Figure 4.10: The impacts of number of reduce tasks.

Figure 4.10 represents the effect of the number of reduce tasks on the job performance. There is trade-off between the performance gain in allocating resources and the overhead incurred in setting up reduce tasks. Initially, increasing the number of reduce tasks utilize the resources better than a large number of reduce tasks that incurs a high overhead in the setting up process, which results in an increased execution time.

The Figure 4.11 shows the impacts for Hadoop by modifying the Java_opts value.

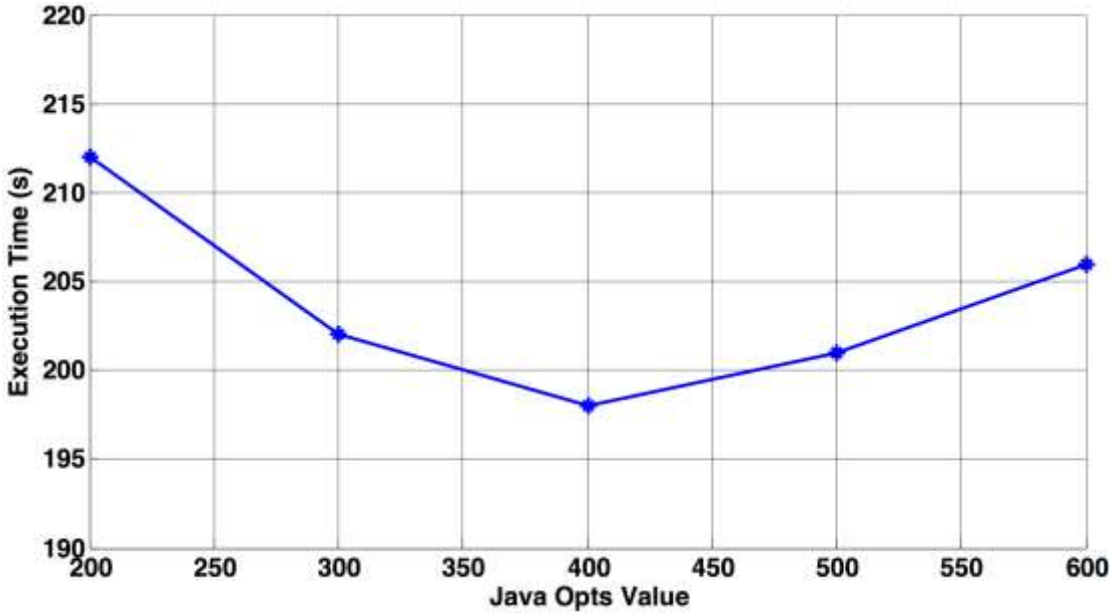


Figure 4.11: The impacts of Java_opts value.

With the value of Java_opts increases, it uses more memory which leads to a decreased execution time. However, when the Java_opts is large enough to consume all the available memory, the job performance will drop dramatically due to the data transfer between the RAM and hard disk.

4.3 Summary

This chapter focuses on experimental analysis. The first experiment is executing K-means clustering algorithm in serial environment and parallel environment severally. The result demonstrates the efficiency of K-means algorithm in serial environment and single-node Hadoop cluster as well as the impacts of different nodes in parallel environment for K-means algorithm. The second experiment illustrates the different efficiencies of running the WordCount application by modifying the parameters on Hadoop. The experiments indicate Hadoop has a plenty of advantages including high scalability, high efficiency and high reliability. In the era of big data, Hadoop and distributed computing will lead the new trend.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Hadoop is a distributed cloud computing platform that can easily analyse and process massive dataset with good performance. Underlying Hadoop is a Hadoop distributed file system (HDFS) that effectively stores dataset. The platform can be deployed on the cheap cluster which is consisted of cheap computers, and it has excellent scalability and flexibility.

Firstly, this thesis particularly introduces and analyses the background and current research status of data mining and cloud computing. Secondly, it discusses the advantages and disadvantages of Hadoop platform, and indicates the MapReduce programming model and the mechanism of Hadoop distributed file system which are two core projects of Hadoop platform. The third chapter introduces the classification of cluster analysis and clustering algorithms, describes K-means clustering algorithm in detail. In addition, it demonstrates the data representation of Mahout and K-means clustering algorithm parallel implementation. Finally, different sizes of datasets are respectively run on serial Weka platform and different number of nodes based on parallel Hadoop platform. The parallel expansibility of Hadoop platform and advantages were compared with serial algorithm by analysing the executing time when dealing with large scale dataset. Meanwhile, it illustrates how the different parameters affect the running of Hadoop platform by analysing the experimental result.

5.2 Future Work

Nowadays, distributed computing and cloud computing are very popular in computing

science, various types of distributed computing platforms appear endlessly, a plenty of IT companies have also introduced some mature products. In the near future, the attempt distributed computing will be fully replaced by distributed. In data mining, increasing data mining algorithms are integrated into distributed platform.

In the future, the Hadoop platform should be further directed to improve its performance and efficiency. In K-means algorithm, the random partition method brings instability to experimental results, some reasonable method should be developed to optimize K-means clustering algorithm. Meanwhile, MapReduce programming can be further optimized, such as the big dataset can be compressed and the small dataset can be merged during the data transfer process. The parallel implementation of other clustering, classification, association rules algorithms in Mahout should gain more attention in the future. In addition, Hadoop configuration parameters have significant impact on the performance of Hadoop clusters; it is able to improve abilities for processing large scale data by modifying Hadoop configuration parameters.

The huge dataset is extensively generated in every industry sector. Merchant is willing to extract the useful information from the transactions in order to make the best decision; researchers are expecting to extract the useful information from the experimental results and thus to develop new theories and products; bankers need to extract useful information from data model to determine the direction of investment. Thus, how to realize the parallel data mining algorithms to improve the executing speed is becoming a significant problem. It requires the efforts from all sectors to achieve the optimum state of data mining.

References

[1] T. White, *Hadoop: The definitive guide*, vol. 54. 2012.

[2] B. J. Gantz and D. Reinsel, "Extracting Value from Chaos State of the Universe : An Executive Summary," *IDC iView*, no. June, pp. 1–12, 2011.

[3] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in *IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 27–33.

[4] S. P. Mirashe and N. V Kalyankar, "Cloud Computing," *Commun. ACM*, vol. 51, no. 7, p. 9, 2010.

[5] B. Furht, A. Escalante, H. Jin, S. Ibrahim, T. Bell, W. Gao, D. Huang, and S. Wu, "Handbook of cloud computing," in *Handbook of Cloud Computing*, no. May, 2010, pp. 3–19.

[6] M. Kantardzic, "Data-Mining Concepts," in *Data Mining*, 2011, pp. 1–25.

[7] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, 2014.

[8] Google, "Google App Engine," *Development*, vol. 2009, pp. 1–10, 2011.

[9] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. p. 29, 2003.

[10] J. Dean and S. Ghemawat, "MapReduce : Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, pp. 1–13, 2008.

[11] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*, 2006, pp. 205–218.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. fifth Berkeley Symp*, vol. 233, no. 233, pp. 281–297, 1967.

[13] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," *KDD Work. text Min.*, vol. 400, pp. 1–2, 2000.

[14] D. Pelleg, D. Pelleg, A. W. Moore, and A. W. Moore, “X-means: Extending K-means with efficient estimation of the number of clusters,” in *Proceedings of the Seventeenth International Conference on Machine Learning table of contents*, 2000, pp. 727–734.

[15] J. Therdphapiyanak and K. Piromsopa, “An analysis of suitable parameters for efficiently applying K-means clustering to large TCPdump data set using Hadoop framework,” in *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2013*, 2013.

[16] Z. Cao and Y. Zhou, “Parallel text clustering based on MapReduce,” in *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*, 2012, pp. 226–229.

[17] Y. Yang, W. Tan, T. Li, and D. Ruan, “Consensus clustering based on constrained self-organizing map and improved Cop-Kmeans ensemble in intelligent decision support systems,” *Knowledge-Based Syst.*, vol. 32, pp. 101–115, 2012.

[18] B. Li, H. Zhao, and Z. Lv, “Parallel ISODATA clustering of remote sensing images based on MapReduce,” in *Proceedings - 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2010*, 2010, pp. 380–383.

[19] R. M. Esteves, R. Pais, and C. Rong, “K-means clustering in the cloud - A Mahout test,” in *Proceedings - 25th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2011*, 2011, pp. 514–519.

[20] D. Borthakur, “The hadoop distributed file system: Architecture and design,” *Hadoop Proj. Website*, pp. 1–14, 2007.

[21] T. White, “Introduction to Hadoop,” *Oracle Mag.*, pp. 40–46, 2014.

[22] J. Dittrich and J. Quian, “Efficient Big Data Processing in Hadoop MapReduce,” *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 2014–2015, 2012.

[23] M. Bhandarkar, “MapReduce programming with apache Hadoop,” *Parallel. Distrib. Process. (IPDPS), 2010 IEEE Int. Symp.*, 2010.

[24] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, “The Performance of MapReduce: An In-depth Study,” *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 472–483, 2010.

[25] L. Ralf, “Google ’ s MapReduce Programming Model — Revisited,” *Sci. Comput. Program.*, vol. 70, no. 1, pp. 1–30, 2008.

[26] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A model of computation for MapReduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms* (SODA '10). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 938-948.

[27] D. Borthakur, “HDFS architecture guide,” *Hadoop Apache Proj. http//hadoop apache ...*, 2008.

[28] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop distributed file system,” in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, 2010.

[29] T. Staff and L. Imagination, “Introducing Apache Mahout,” 2009.

[30] R. X. and D. C. W. II, *Clustering*. 2009.

[31] J. Han and M. Kamber, *Data mining : concepts and techniques*. 2001.

[32] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “An efficient k-means clustering algorithm: analysis and implementation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, 2002.

[33] D. T. Pham, S. S. Dimov, and C. D. Nguyen, “Selection of K in K-means clustering,” *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 219, no. 1, pp. 103–119, 2005.

[34] M. Yamamoto and Y. Terada, “Functional factorial K-means analysis,” *Comput. Stat. Data Anal.*, vol. 79, pp. 133–148, 2014.

[35] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. 2011.

[36] A. McCallum, K. Nigam, and L. H. Ungar, “Efficient clustering of high-dimensional data sets with application to reference matching,” *Proc. sixth ACM SIGKDD Int. Conf. Knowl. Discov. data Min. KDD 00*, pp. 169–178, 2000.

[37] A. Aizawa, “An information-theoretic perspective of tf-idf measures,” *Inf. Process. Manag.*, vol. 39, no. 1, pp. 45–65, 2003.

[38] J. Ramos, J. Eden, and R. Edu, “Using TF-IDF to Determine Word Relevance in Document Queries,” *Processing*, 2003.

[39] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, “Interpreting TF-IDF term weights as making relevance decisions,” *ACM Transactions on Information Systems*, vol. 26, no. 3. pp. 1–37, 2008.

[40] W. Zhao, H. Ma, and Q. He, “Parallel K-means clustering based on MapReduce,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5931 LNCS, pp. 674–679.

[41] Hadoop Wiki (2009) *SequenceFile*. Availiable at: http://wiki.apache.org/hadoop/SequenceFile.(Accessed: 30 July 2015)

[42] G. Holmes; A. Donkin; I.H. Witten. "Weka: A machine learning workbench" (PDF). *Proc Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia*, 1994.

[43] K. Wang, X. Lin, and W. Tang, “Predator-An experience guided configuration optimizer for Hadoop MapReduce,” in *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*, 2012, pp. 419–426.